大学数学 (类) 学科期末备考建议

卢鹏博[1] 3314982394@qq.com

摘要

在大学学习中，我们会遇到许许多多的期末考试，然而有时我们精力全身心投入学习，导致期末前许多人紧张。本文从多个角度出发分析数学（类）学科期末备考的时间充裕度，并从多方面证明了一星期备考基本足够的结论。

关键词：大学学习，期末考试, 数学考试

Abstract

During university studies, we encounter numerous final exams. However, sometimes our full dedication to studying causes many to feel nervous before the exams.This article analyzes the sufficiency of time for preparing for final exams in mathematics (or related) disciplines from multiple perspectives, and demonstrates from various aspects that one week of preparation is generally sufficient.

Keywords：University studies, **Final exams**, **Mathematic exams**

一、 引言

最近陆陆续续到了大学的考试周，许多大学生都开始准备期末考试了。有一部分社会人士认为大学的期末考试很简单，基本不需要备考。但是大学生们不同意，他们想要考过好成绩带回家。确实，期末考试是大学生放假前的最后一道门槛，如果能够考个好成绩，那就能快快乐乐度过这个暑假。本文将着重分析大学生最为头疼的学科之一——数学学科的备考安排与建议。

二、 考题预测分析

不同于期中考试，期末考试的题目会更加简单。有些学校老师也要保证及格率 [1]，不会为难学生。另外，有的老师在期末考试之前也会划重点，这大大缩小了复习范围。因此我们从学科难度，题目难度与考题重点三部分分类叙述。

[1]MathEnthusiast，中国

2.1 学科难度

本文主要讨论数学类别的期末考试（其它理科类似），因此在课程难度上将肯定是比较高的。

数学的教程一般分两类，一类是理论基础的学科，比如：微积分，线性代数，数学分析等。这些看似比较抽象，不那么具体，反而更加简单。具体原因有三点：

1. 题目方法基本固定：不像一些建模等类型的题目，方法较多，做题时需要尝试并选择一个合适的方法，因此比较耗时间；

2. 公式较少容易记忆：由于都是基本内容，因此对应的公式等都比较简单，困难的公式不多。而且都是经典的结论，因此公式都比较简洁好用；

3. 计算量小不易出错：这类题基本上都是按步给分的，不会因为结果错误就不给分。而且由于公式都比较简洁，因此计算上也不容易出错。

另一类是建模类的学科，比如：数学建模，数理统计，微（宏）观经济学等。这类题虽然是解决实际问题，但是只看公式是很难看懂的，因此这类学科需要辅助例题一起复习。当然，这类学科在复习上会有优势，因为更加具体，只要弄懂一道题，其它类似的题基本上就会做了。

2.2 题目难度

期末考试一般不难。

首先，期末考试的难度一般是比期中考试简单的。而期中考试，大多是一些基础和中等题，难题偏少。因此由此推测期末基本上是基础题和中等题，难题 0-1 道（特殊情况除外）。

其次，正如前面所说，期末有及格率要求。因此老师一般不会把题目设置过难，导致及格率过低。

最后，期末考题选题有可能是书本的例题或者课后题，因此熟练掌握书本上的习题，那样期末考试就没什么难度了。

当然，考试题目也不能过于简单，如果全部都 90 分以上，老师也有可能受到批评的。因此不能寄希望于“裸考”，除非有天赋或者学过相关内容，否则最多只能及格。

2.3 考题重点

有些老师会在课程最后划重点，这就大大缩小了考试的范围，更容易复习！

三、 备考分析

下面我将从程序模拟，个人实际验证两部分探讨备考时间。

3.1 模型预测

我们将使用 Python 程序模拟学生的复习及成绩预测。为了简化模型，我们首先对模型提出以下六点假设：

模型假设

1. 考试一共 10 道题，每题 10 分。每题考两个知识点（板块），一共 15 个知识点。每天学习 (复习)4-5 个知识点
2. 知识点的熟练度和题目正确率取值均为 $[0,1]$
3. 如果某一知识点从未学习过，那么学习一次的熟练度为 (0.5,0.8) 的随机值
4. 如果一个知识点学过，那么复习一次的熟练度**提升** 0.4*(1-熟练度)
5. 每一天模拟 1000 次，以这 1000 次的成绩的数学期望作为预测成绩
6. 以时间 t 作为一个时间变量，分别计算 0-14 天的预测成绩

表 **1**　参数说明

参数名	参数值	单位
num_questions	10	题
knowledge_per_question	2	个
total_knowledge_points	15	个
min_learn_per_day	4	个
max_learn_per_day	5	个
initial_learn_gain	0.3	-
review_boost	0.4	-
simulations_per_t	1000	次
max_days	14	天

为了区别不同程度的学生，我们分为 4 组学生，分别为：零基础（基础熟练度全为 0），较生疏（基础熟练度为 0.3-0.5 的随机数），比较熟练（基础熟练度为 0.5-0.7 的随机数），非常熟练（基础熟练度为 0.7-0.9 的随机数）四类。

我们首先进行第一次模拟，这次模拟是非常简单的，假设的也非常极端（零基础的熟练度是 0，实际上一般不会是 0），以及假设学生只复习了要考的知识点。

通过计算机模拟，得到成绩——学习时间曲线图如下：

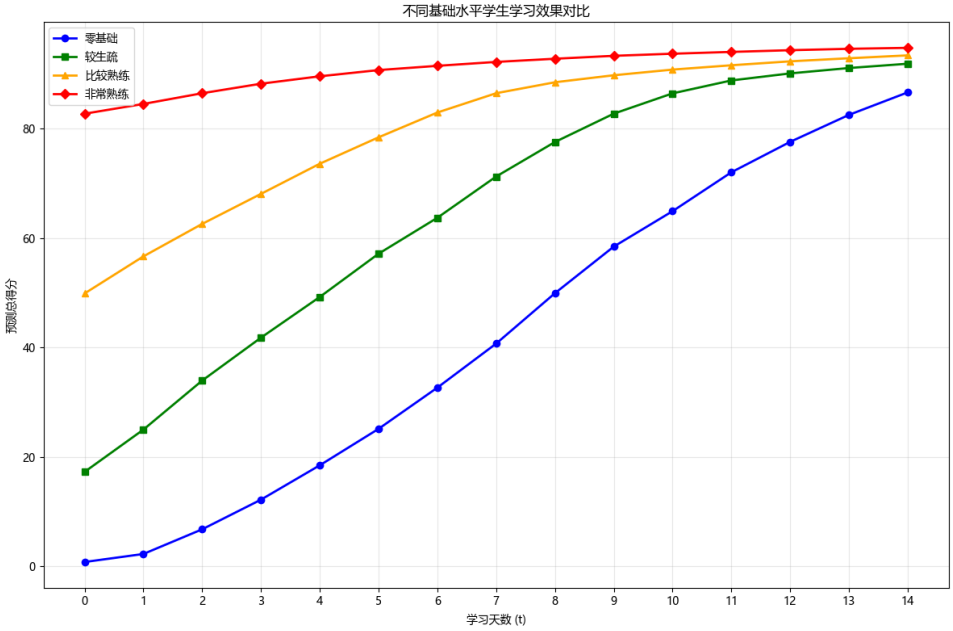

图 **1**　不同基础水平学生学习效果对比

从该图可以看到在完全零基础的情况下，学生从复习 9 天左右即可达到及格，13 天即可达到 80 分。如果有一点基础（上课听过），那么复习 6 天即可达到及格水平，复习 9 天即可达到 80 分。如果比较熟练（上课听了 + 作业做了），复习 2 天左右即可达到及格水平，复习 6 天左右即可达到 80 分。如果非常熟练，那么复习提升（百分比）并不明显，复习一周可提升至 90 分。

第一次模拟考虑的比较简单，第二次在第一次的基础上，加上考场不会的新题（此时题目成绩记为 0-2 分的随机数，而不是数学期望计算分数），以及灵机一动注意到了解法（此时题目记为 10 分，而不是数学期望计算分数）。发生这两种情况的概率均设为 1%，当发生这两种情况之一时，以本次的成绩（不是数学期望）作为一个特殊情况，同时不改变原有的 1000 次预测及的规则，将所以特殊情况整合在一起计算成绩的数学期望作为特殊情况的成绩预测，特殊情况的图与其他图分开画。另外，加上可能用不到的知识点，比如复习了 20-30 个知识点，但只考了 15 个。此外重新定义零基础（基础熟练度为 0.4），较生疏（基础熟练度为 0.4-0.6 的随机数），比较熟练（基础熟练度为 0.6-0.75 的随机数），非常熟练（基础熟练度为 0.75-0.95 的随机数）四类。

当加上“无用”（不考）的知识点时，学生的学习效率（相比之前）普遍变慢了。因此我调整了一共 25 个知识点，考 15 个，学生每天学 5-8 个知识点。下图分别是预测学习成绩以及特殊情况次数。

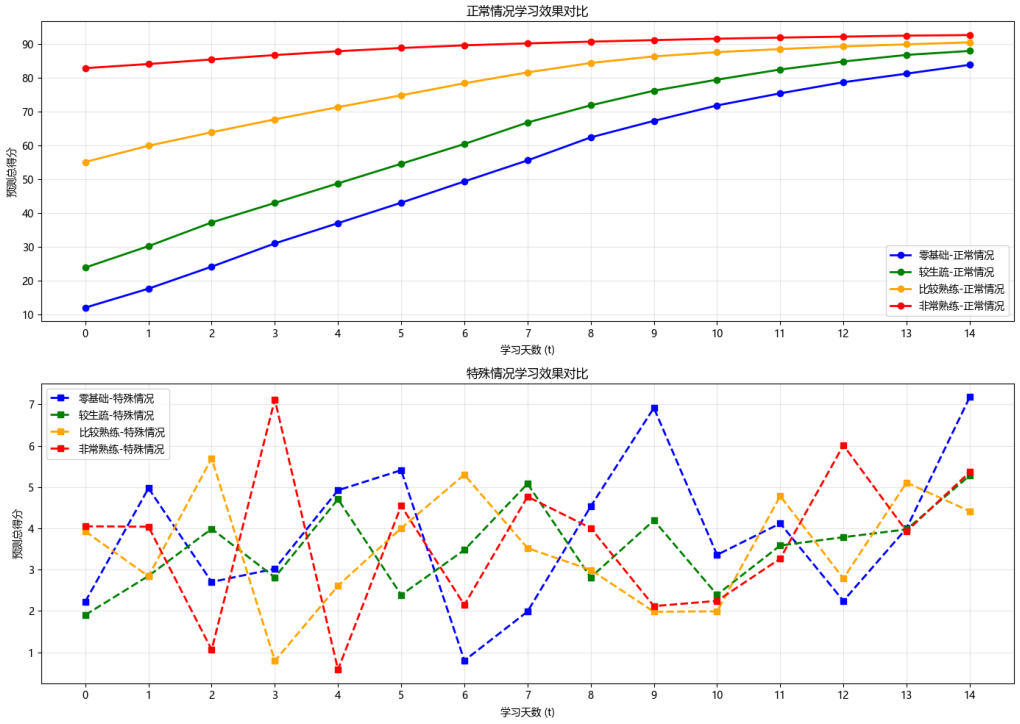

图 **2**　第二次模拟不同基础水平学生学习效果对比

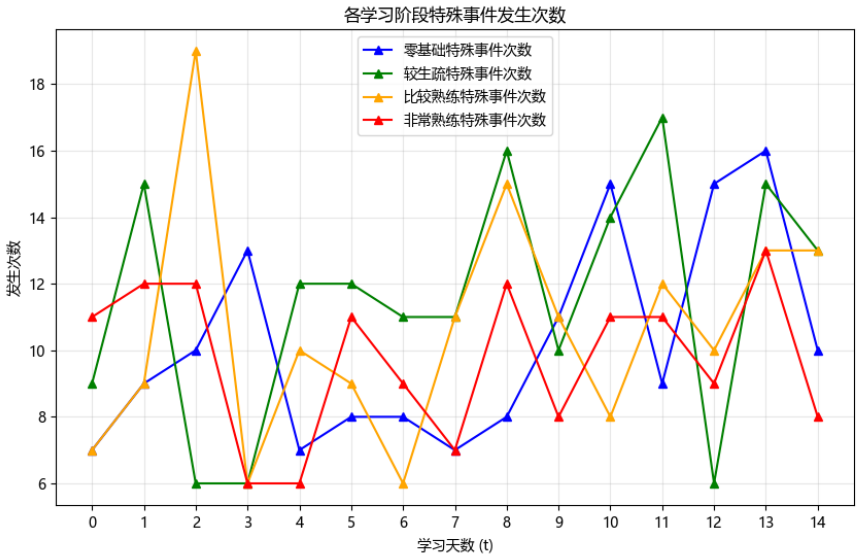

图 **3**　特殊情况发生次数

由图 2 可以看出“零基础”的学生只要学习 8 天即可达到及格，比较熟悉的学习 7 天可以达到 80 分。

在特殊情况（完全不会或灵机一动）的情况下，学生的的评价成绩为 5 分。而且特殊情况发生的概率不大（每一天模拟 1000 次，图 3 为特殊情况发生的次数）。

当然，第二次模拟也少考虑了一些东西，比如对知识点越熟练，灵机一动的概率越大，反之越小。当然，现实中不会做的概率应该比灵机一动的概率大，因此特殊情况的成绩的均值应该小于 5。另外，正确率曲线（前段增长较慢）也有点保守，因此我们继续改进代码。

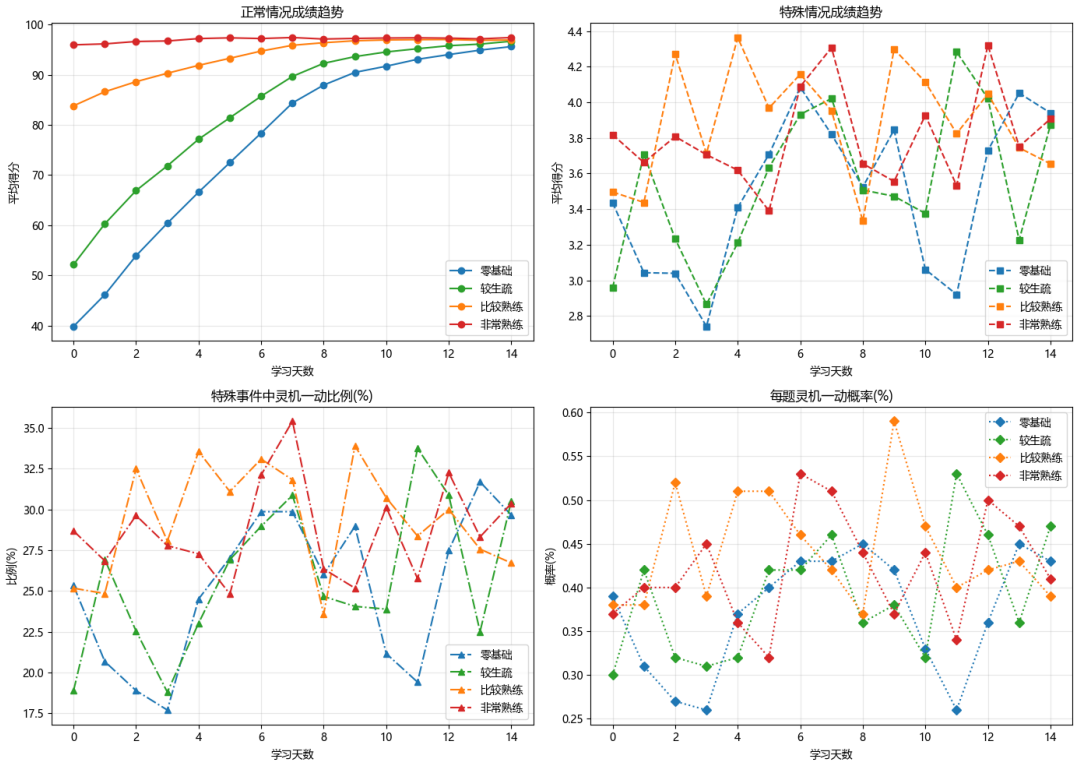

图 **4**　第三次模拟不同基础水平学生学习效果及突发事件汇总

通过图片可以看出，即使“零基础”，只要认真学 3 天左右，还是能及格的，学 7 天左右可以达到 80 分。对于较生疏的学生，认真学习 1 天即可及格，学习 5 天可以达到 80 分。

3.2 个人预测

以本人为例，曾经考过的几门数学科目，比如《概率论》，《运筹学》，《高等代数》，《大学物理》四门学科。以 L0-L3 分别代表零基础-非常熟练初始熟练度分别为：L1.5（介于 L1 和 L2），L1.5，L2，L1，我分别复习了 7 天，7 天，10 天，7 天，分数分别为：99,96,98,89。**基本符合预测图**。

四、 结论与建议

五、 结论

通过使用 Python 程序的三次模拟，以及个人实际的验证，可以说“零基础”备考期末一星期足够了（不保证适用于每个人）。

六、 建议

由于基础不同，所以所需要的复习时间不同。

对于“零基础”和比较生疏的学生，建议认真复习至少一星期，复习过程中要结合书上的例题和课后习题，先把定义和简单的题弄懂，有精力再攻克难题。不要过于指望考试时能够“灵机一动”。

对于比较熟练和非常收敛的学生，此时对课本的定义定理和题目比较熟练。建议再把定义定理过一遍，以加深印象。对于课后题，可以思考从多个角度去解决，增强考场应变能力。

参考文献

[1] 知乎.【大学规定！学生挂科率超 50%，教师考核不合格】[EB/OL].(2025-2-14)[2025-6-10].https://www.zhihu.com/question/457765989

附录 A　代码及图片

见文件夹 CAF (codes and figures) 中的：

First.py, 1.png

Second.py, 2_1.png, 2_2.png

Third.py, 3.png

附录 B　第一次预测 Python 代码

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']

# ========== 参数设置 ==========
```

```
num_questions = 10
knowledge_per_question = 2
total_knowledge_points = 15

# 学习参数
min_learn_per_day = 4
max_learn_per_day = 5
initial_learn_gain = 0.3 # 初始学习增益改为固定值
review_boost = 0.4

# 模拟参数
simulations_per_t = 1000
max_days = 14

# 学生组别设置（改为初始熟练度范围）
student_groups = {
    "零基础": (0, 0),
    "较生疏": (0.3, 0.5),
    "比较熟练": (0.5, 0.7),
    "非常熟练": (0.7, 0.9)
}


# ========== 改进的正确率函数 ==========
def calculate_accuracy(mastery):
    """更平滑的S型曲线，避免突变"""
    return 1 / (1 + np.exp(-8 * (mastery - 0.6)))


# ========== 改进的模拟函数 ==========
def simulate_score(days, initial_range):
    total_scores = 0

    for _ in range(simulations_per_t):
        # 初始化熟练度
        if initial_range == (0, 0):
            mastery = np.zeros(total_knowledge_points)
        else:
            mastery = np.random.uniform(initial_range[0], initial_range[1],
                total_knowledge_points)

        learn_count = np.zeros(total_knowledge_points, dtype=int)

        for day in range(days):
            learn_today = np.random.randint(min_learn_per_day, max_learn_per_day + 1)

            # 改进的学习策略：优先复习不熟练的知识点
```

```
            needs_review = np.where((learn_count > 0) & (mastery < 0.8))[0]
            new_topics = np.where(learn_count == 0)[0]

            # 平衡新学和复习
            if len(needs_review) > 0:
                review_num = min(learn_today // 2 + 1, len(needs_review))
                chosen_review = np.random.choice(needs_review, review_num, replace=False)
                remain = learn_today - review_num
            else:
                chosen_review = np.array([], dtype=int)
                remain = learn_today

            if remain > 0 and len(new_topics) > 0:
                chosen_new = np.random.choice(new_topics, min(remain, len(new_topics)),
                    replace=False)
                chosen = np.concatenate([chosen_review, chosen_new])
            else:
                chosen = chosen_review

            # 确保选择足够数量的知识点
            if len(chosen) < learn_today:
                additional = np.random.choice(total_knowledge_points, learn_today -
                    len(chosen), replace=False)
                chosen = np.unique(np.concatenate([chosen, additional]))
                chosen = chosen[:learn_today]

            # 改进的学习效果计算
            for point in chosen:
                if learn_count[point] == 0: # 新学
                    mastery[point] += initial_learn_gain * (1 - mastery[point])
                else: # 复习
                    mastery[point] += review_boost * (1 - mastery[point])
                learn_count[point] += 1

        # 计算成绩
        question_probs = []
        for _ in range(num_questions):
            k1, k2 = np.random.choice(total_knowledge_points, 2, replace=True)
            avg_mastery = (mastery[k1] + mastery[k2]) / 2
            question_probs.append(calculate_accuracy(avg_mastery))

        total_scores += np.sum(question_probs) * 10

    return total_scores / simulations_per_t


# ========== 运行模拟 ==========
```

```
print("天数\t零基础\t较生疏\t比较熟练\t非常熟练")
print("----------------------------------------")
days_range = np.arange(0, max_days + 1)
all_scores = {}

for group_name, initial_range in student_groups.items():
    scores = [simulate_score(t, initial_range) for t in days_range]
    all_scores[group_name] = scores
    print(f"{group_name}组模拟完成")

# 打印结果
for i, t in enumerate(days_range):
    print(
        f"{t}\t{all_scores['零基础'][i]:.2f}\t{all_scores['较生疏'][i]:.2f}
        \t{all_scores['比较熟练'][i]:.2f}\t{all_scores['非常熟练'][i]:.2f}")

# ========== 绘图 ==========
plt.figure(figsize=(12, 8))
colors = ['blue', 'green', 'orange', 'red']
markers = ['o', 's', '^', 'D']

for i, (group_name, scores) in enumerate(all_scores.items()):
    plt.plot(days_range, scores, color=colors[i], marker=markers[i],
             label=f'{group_name}', linewidth=2)

plt.title('不同基础水平学生学习效果对比')
plt.xlabel('学习天数 (t)')
plt.ylabel('预测总得分')
plt.xticks(days_range)
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()
```

附录 C　第二次预测 Python 代码

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.rcParams['axes.unicode_minus'] = False

# ========== 参数设置 ==========
num_questions = 10
```

```
knowledge_per_question = 2
total_knowledge_points = 25 # 总知识点数（实际考试只用部分）
exam_knowledge_points = 15 # 考试涉及的知识点数

# 学习参数
min_learn_per_day = 5
max_learn_per_day = 8
initial_learn_gain = 0.3
review_boost = 0.4

# 特殊事件概率
p_special = 0.01 # 特殊事件总概率
p_blank = 0.7 # 遇到特殊事件时，不会做的概率
p_inspiration = 0.3 # 遇到特殊事件时，灵机一动的概率

# 模拟参数
simulations_per_t = 1000
max_days = 14

# 重新定义学生组别
student_groups = {
    "零基础": (0.4, 0.4), # 固定0.4
    "较生疏": (0.4, 0.6), # 0.4-0.6随机
    "比较熟练": (0.6, 0.75), # 0.6-0.75随机
    "非常熟练": (0.75, 0.95) # 0.75-0.95随机
}


# ========== 改进的正确率函数 ==========
def calculate_accuracy(mastery):
    """更平滑的S型曲线"""
    return 1 / (1 + np.exp(-8 * (mastery - 0.65)))


# ========== 改进的模拟函数 ==========
def simulate_score(days, initial_range):
    normal_scores = []
    special_scores = []

    for _ in range(simulations_per_t):
        # 初始化熟练度
        if initial_range[0] == initial_range[1]:
            mastery = np.full(total_knowledge_points, initial_range[0])
        else:
            mastery = np.random.uniform(initial_range[0], initial_range[1],
                total_knowledge_points)
```

```
    learn_count = np.zeros(total_knowledge_points, dtype=int)
    exam_knowledge = np.random.choice(total_knowledge_points, exam_knowledge_points,
        replace=False)

    for day in range(days):
        learn_today = np.random.randint(min_learn_per_day, max_learn_per_day + 1)

        # 改进的学习策略
        needs_review = np.where((learn_count > 0) & (mastery < 0.85))[0]
        new_topics = np.where(learn_count == 0)[0]

        # 平衡新学和复习
        review_num = min(learn_today // 2 + 1, len(needs_review))
        chosen_review = np.random.choice(needs_review, review_num, replace=False) if
            len(
            needs_review) > 0 else np.array([], dtype=int)
        remain = learn_today - review_num

        if remain > 0 and len(new_topics) > 0:
            chosen_new = np.random.choice(new_topics, min(remain, len(new_topics)),
                replace=False)
            chosen = np.concatenate([chosen_review, chosen_new])
        else:
            chosen = chosen_review

        # 确保选择足够数量的知识点
        if len(chosen) < learn_today:
            additional = np.random.choice(total_knowledge_points, learn_today -
                len(chosen), replace=False)
            chosen = np.unique(np.concatenate([chosen, additional]))
            chosen = chosen[:learn_today]

        # 更新熟练度
        for point in chosen:
            if learn_count[point] == 0: # 新学
                mastery[point] += initial_learn_gain * (1 - mastery[point])
            else: # 复习
                mastery[point] += review_boost * (1 - mastery[point])
            learn_count[point] += 1

    # 计算正常情况成绩
    normal_probs = []
    for _ in range(num_questions):
        k1, k2 = np.random.choice(exam_knowledge, 2, replace=True)
        avg_mastery = (mastery[k1] + mastery[k2]) / 2
        normal_probs.append(calculate_accuracy(avg_mastery))
    normal_score = np.sum(normal_probs) * 10
```

```
            # 处理特殊情况
            special_score = None
            if np.random.rand() < p_special:
                if np.random.rand() < p_blank: # 不会做的情况
                    special_score = np.random.uniform(0, 2)
                else: # 灵机一动的情况
                    special_score = 10.0

            normal_scores.append(normal_score)
            if special_score is not None:
                special_scores.append(special_score)

        # 计算平均成绩
        avg_normal = np.mean(normal_scores) if normal_scores else 0
        avg_special = np.mean(special_scores) if special_scores else avg_normal

        return {
            "normal": avg_normal,
            "special": avg_special,
            "special_count": len(special_scores)
        }


# ========== 运行模拟 ==========
print("模拟进度:")
days_range = np.arange(0, max_days + 1)
all_results = {}

for group_name, initial_range in student_groups.items():
    group_results = {"normal": [], "special": [], "special_count": []}
    for t in days_range:
        result = simulate_score(t, initial_range)
        group_results["normal"].append(result["normal"])
        group_results["special"].append(result["special"])
        group_results["special_count"].append(result["special_count"])
    all_results[group_name] = group_results
    print(f"{group_name}组模拟完成")

# ========== 结果展示 ==========
print("\n天数\t零基础(常)\t零基础(特)\t较生疏(常)\t较生疏(特)
\t比较熟练(常)\t比较熟练(特)\t非常熟练(常)\t非常熟练(特)")
print("=" * 120)
for i, t in enumerate(days_range):
    row = [t]
    for group in student_groups.keys():
        row.append(f"{all_results[group]['normal'][i]:.2f}")
```

```
            row.append(f"{all_results[group]['special'][i]:.2f}")
        print("\t".join(map(str, row)))

    # ========== 绘图 ==========
    plt.figure(figsize=(14, 10))

    # 正常情况图表
    plt.subplot(2, 1, 1)
    colors = ['blue', 'green', 'orange', 'red']
    for i, group_name in enumerate(student_groups.keys()):
        plt.plot(days_range, all_results[group_name]["normal"], color=colors[i],
                 marker='o', label=f'{group_name}-正常情况', linewidth=2)
    plt.title('正常情况学习效果对比')
    plt.xlabel('学习天数 (t)')
    plt.ylabel('预测总得分')
    plt.xticks(days_range)
    plt.grid(True, alpha=0.3)
    plt.legend()

    # 特殊情况图表
    plt.subplot(2, 1, 2)
    for i, group_name in enumerate(student_groups.keys()):
        plt.plot(days_range, all_results[group_name]["special"], color=colors[i],
                 marker='s', linestyle='--', label=f'{group_name}-特殊情况', linewidth=2)
    plt.title('特殊情况学习效果对比')
    plt.xlabel('学习天数 (t)')
    plt.ylabel('预测总得分')
    plt.xticks(days_range)
    plt.grid(True, alpha=0.3)
    plt.legend()

    plt.tight_layout()
    plt.show()

    # 绘制特殊事件发生次数
    plt.figure(figsize=(10, 6))
    for i, group_name in enumerate(student_groups.keys()):
        plt.plot(days_range, all_results[group_name]["special_count"], color=colors[i],
                 marker='^', label=f'{group_name}特殊事件次数')
    plt.title('各学习阶段特殊事件发生次数')
    plt.xlabel('学习天数 (t)')
    plt.ylabel('发生次数')
    plt.xticks(days_range)
    plt.grid(True, alpha=0.3)
    plt.legend()
    plt.show()
```

附录 D　第三次预测 Python 代码

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.rcParams['axes.unicode_minus'] = False

# ========== 参数设置 ==========
num_questions = 10
knowledge_per_question = 2
total_knowledge_points = 25
exam_knowledge_points = 15

# 学习参数
min_learn_per_day = 5
max_learn_per_day = 8
initial_learn_gain = 0.35
review_boost = 0.45

# 特殊事件概率
p_special = 0.015 # 特殊事件基础概率

# 模拟参数
simulations_per_t = 1000
max_days = 14

# 学生组别定义
student_groups = {
    "零基础": (0.4, 0.4),
    "较生疏": (0.4, 0.6),
    "比较熟练": (0.6, 0.75),
    "非常熟练": (0.75, 0.95)
}


# ========== 改进的正确率函数 ==========
def calculate_accuracy(mastery):
    if mastery<=0.5:
        return mastery
    else:
        return 1 / (1 + np.exp(-10 * (mastery - 0.5)))

# ========== 动态灵机一动概率函数 ==========
def inspiration_probability(avg_mastery):
```

```
    """基于熟练度的非线性概率"""
    return 0.05 + 0.25 * (1 - np.exp(-3 * avg_mastery)) # 保证概率在5%-30%之间


# ========== 模拟函数 ==========
def simulate_score(days, initial_range):
    normal_scores = []
    special_scores = []
    special_counts = {"inspire": 0, "blank": 0}

    for _ in range(simulations_per_t):
        # 初始化熟练度
        mastery = np.full(total_knowledge_points, initial_range[0]) if initial_range[0]
            == initial_range[1] \
            else np.random.uniform(initial_range[0], initial_range[1],
                total_knowledge_points)

        learn_count = np.zeros(total_knowledge_points, dtype=int)
        exam_knowledge = np.random.choice(total_knowledge_points, exam_knowledge_points,
            replace=False)

        # 学习过程
        for day in range(days):
            learn_today = np.clip(np.random.normal((min_learn_per_day +
                max_learn_per_day) / 2, 1),
                            min_learn_per_day, max_learn_per_day).astype(int)

            # 动态选择学习内容
            needs_review = np.where((learn_count > 0) & (mastery < 0.85))[0]
            new_topics = np.where(learn_count == 0)[0]

            review_num = min(int(learn_today * 0.6), len(needs_review))
            chosen = list(np.random.choice(needs_review, review_num, replace=False)) if
                review_num > 0 else []
            chosen += list(np.random.choice(new_topics, min(learn_today - review_num,
                len(new_topics)), replace=False))

            # 更新熟练度
            for point in chosen:
                if learn_count[point] == 0:
                    mastery[point] = min(1,mastery[point] + initial_learn_gain * (1 -
                        mastery[point]) * np.random.uniform(0.8, 1.2))
                else:
                    mastery[point] = min(1, mastery[point] + review_boost * (1 -
                        mastery[point]) * np.random.uniform(0.8,1.2))
                learn_count[point] += 1
```

```
        # 考试模拟
        question_scores = []
        inspire_count = 0
        blank_count = 0

        for _ in range(num_questions):
            k1, k2 = np.random.choice(exam_knowledge, 2, replace=True)
            avg_mastery = (mastery[k1] + mastery[k2]) / 2

            # 处理特殊事件
            if np.random.rand() < p_special:
                p_inspire = inspiration_probability(avg_mastery)
                if np.random.rand() < p_inspire:
                    score = 10.0
                    inspire_count += 1
                else:
                    score = np.random.uniform(0, 2) * (1 - avg_mastery)
                    blank_count += 1
            else:
                score = calculate_accuracy(avg_mastery) * 10

            question_scores.append(score)

        # 统计结果
        normal_score = np.sum([s for s in question_scores if s <= 10]) # 正常得分
        special_score = inspire_count * 10 + blank_count * 1 # 特殊事件总得分

        normal_scores.append(normal_score)
        if inspire_count + blank_count > 0:
            special_scores.append(special_score)
            special_counts["inspire"] += inspire_count
            special_counts["blank"] += blank_count

    # 计算结果
    avg_normal = np.mean(normal_scores) if normal_scores else 0
    avg_special = np.mean(special_scores) if special_scores else 0

    total_special = special_counts["inspire"] + special_counts["blank"]
    inspire_rate = special_counts["inspire"] / total_special if total_special > 0 else 0

    return {
        "normal": avg_normal,
        "special": avg_special,
        "special_count": len(special_scores),
        "inspire_rate": inspire_rate,
        "inspire_per_question": special_counts["inspire"] / (
                    simulations_per_t * num_questions) if simulations_per_t > 0 else 0
```

```
    }


# ========== 运行模拟 ==========
print("模拟进度:")
days_range = np.arange(0, max_days + 1)
all_results = {}

for group_name, initial_range in student_groups.items():
    group_results = {
        "normal": [],
        "special": [],
        "special_count": [],
        "inspire_rate": [],
        "inspire_per_question": []
    }

    for t in days_range:
        result = simulate_score(t, initial_range)
        group_results["normal"].append(result["normal"])
        group_results["special"].append(result["special"])
        group_results["special_count"].append(result["special_count"])
        group_results["inspire_rate"].append(result["inspire_rate"])
        group_results["inspire_per_question"].append(result["inspire_per_question"])
        print(f"{group_name} 第{t}天模拟完成", end="\r")

    all_results[group_name] = group_results
    print(f"{group_name}组模拟完成{' ' * 30}")


# ========== 可视化结果 ==========
def plot_results():
    colors = ['#1f77b4', '#2ca02c', '#ff7f0e', '#d62728']

    # 正常情况成绩
    plt.figure(figsize=(14, 10))
    plt.subplot(2, 2, 1)
    for i, group_name in enumerate(student_groups.keys()):
        plt.plot(days_range, all_results[group_name]["normal"],
                 color=colors[i], marker='o', label=group_name)
    plt.title('正常情况成绩趋势')
    plt.xlabel('学习天数')
    plt.ylabel('平均得分')
    plt.grid(True, alpha=0.3)
    plt.legend()

    # 特殊情况成绩
```

```
    plt.subplot(2, 2, 2)
    for i, group_name in enumerate(student_groups.keys()):
        plt.plot(days_range, all_results[group_name]["special"],
                 color=colors[i], marker='s', linestyle='--', label=group_name)
    plt.title('特殊情况成绩趋势')
    plt.xlabel('学习天数')
    plt.ylabel('平均得分')
    plt.grid(True, alpha=0.3)
    plt.legend()

    # 灵机一动概率
    plt.subplot(2, 2, 3)
    for i, group_name in enumerate(student_groups.keys()):
        plt.plot(days_range, np.array(all_results[group_name]["inspire_rate"]) * 100,
                 color=colors[i], marker='^', linestyle='-.', label=group_name)
    plt.title('特殊事件中灵机一动比例(%)')
    plt.xlabel('学习天数')
    plt.ylabel('比例(%)')
    plt.grid(True, alpha=0.3)
    plt.legend()

    # 每题灵机一动概率
    plt.subplot(2, 2, 4)
    for i, group_name in enumerate(student_groups.keys()):
        plt.plot(days_range, np.array(all_results[group_name]["inspire_per_question"]) *
            100,
                 color=colors[i], marker='D', linestyle=':', label=group_name)
    plt.title('每题灵机一动概率(%)')
    plt.xlabel('学习天数')
    plt.ylabel('概率(%)')
    plt.grid(True, alpha=0.3)
    plt.legend()

    plt.tight_layout()
    plt.show()


plot_results()

# 打印关键数据
print("\n关键统计数据:")
print("组别\t最终正常分\t最终特殊分\t灵机一动率\t每题灵机概率")
for group_name in student_groups.keys():
    print(f"{group_name}\t{all_results[group_name]['normal'][-1]:.1f}\t\t"
          f"{all_results[group_name]['special'][-1]:.1f}\t\t"
          f"{all_results[group_name]['inspire_rate'][-1] * 100:.1f}%\t\t"
          f"{all_results[group_name]['inspire_per_question'][-1] * 100:.2f}%")
```