2026 MCM　A：智能手机电池消耗建模

这典型的连续系统仿真 问题。不可以直接用简单的机器学习（LSTM/Transformer）去跑黑盒预测。题目明确要求连续时间描述,我们的模型必须是微分方程（ODEs）。

问题一：建立连续时间模型

题目要求： 开发一个连续时间数学模型，描述电池荷电状态 (SOC) 随时间的变化，需考虑屏幕、CPU、网络、后台任务以及环境温度 。

问题抽象：

类似的 “水池模型”（Inflow - Outflow = Storage Change）。对于电池而言，主要是 Outflow（放电）。我们需要建立 $SOC(t)$*关于时间*t 的导数方程。

模型构建推荐方案：

1. 核心方程： 使用常微分方程（ODE）。

$$\frac{d(SOC)}{dt} = -\frac{1}{C_{eff}(T,\text{Age})} \cdot I_{total}(t) \tag{1}$$

其中C_{eff} 是有效容量，I_{total} 是总负载电流。

2. 电流分解（关键得分点）： 不能把 I_{total} 当成一个常数，它是一个动态的 复合函数

$$I_{total}(t) = I_{base} + I_{screen}(Brightness, Area) + I_{cpu}(Load) + I_{network}(Signal) \tag{2}$$

引入 非线性修正： 引入 Peukert 效应（大电流放电时容量会“缩水”）和 Arrhenius 方程（温度对化学反应速率的影响）。

```
Simulation Results (Time-to-Empty):
Gaming: 0.03 hours
Video: 4.67 hours
Reading: 15.00 hours
```

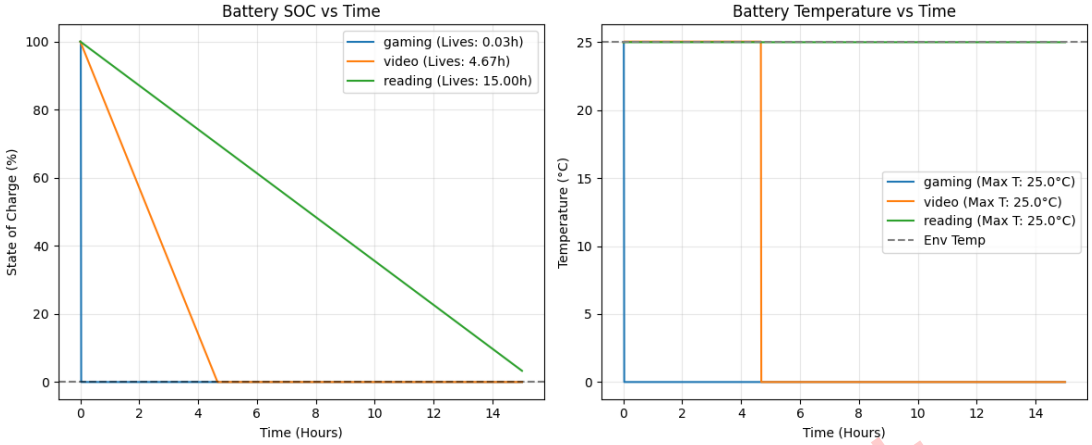

Figure 1. Preliminary Model Results

Python 代码 (odeint 求解器):

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
# 1. 定义物理参数
C_design = 4000  # mAh (设计容量)
ref_temp = 298.15 # 25 摄氏度 (K)

# 2. 辅助函数：计算不同组件的电流 (mA)
def get_current_components(t, scenario):
    # 基础待机电流
    i_base = 50
    # 屏幕电流 (假设与亮度呈指数或线性关系)
    if scenario == 'gaming':
        brightness = 0.8 # 80% 亮度
        i_screen = 400 * brightness
        i_cpu = 1500 + 200 * np.sin(t) # 游戏时 CPU 波动大
```

```
    elif scenario == 'reading':
        brightness = 0.3
        i_screen = 400 * brightness
        i_cpu = 100 #  仅文本渲染
    else:
        i_screen = 0
        i_cpu = 20
    return i_base + i_screen + i_cpu
# 3.  核心微分方程  d(SOC)/dt
def battery_dynamics(soc, t, temp, cycles, scenario):
    if soc <= 0: return 0 #  电量耗尽
    #  温度修正系数  (Arrhenius-like)
    k_temp = 1.0 - 0.005 * abs(temp - 25)

    #  老化修正系数
    k_aging = 1.0 - (cycles * 0.0002)
    #  当前有效容量
    c_effective = C_design * k_temp * k_aging
    #  获取瞬时电流
    current_ma = get_current_components(t, scenario)
    #  核心方程: dSOC/dt = - I / C
    #  注意单位: mA / mAh = 1/h
    dsoc_dt = - current_ma / c_effective
    return dsoc_dt
# 4.  求解
t = np.linspace(0, 24, 1000) #  模拟 24 小时
initial_soc = 1.0 # 100%
```

```
temp_val = 25 # 25 度

cycles_val = 100 #  循环 100 次

#  模拟  "Gaming"  场景

sol = odeint(battery_dynamics, initial_soc, t, args=(temp_val, cycles_val, 'gaming'))
```

问题二：预测 Time-to-Empty (Prediction)

题目要求： 预测不同场景下的耗尽时间，并量化不确定性 。

思路分析：

这不仅仅是求 $SOC = 0$ 的时间点，重点在于比较和归因。需要先定义几个典型的 用户画像：

- Profile A (The Gamer): 高 CPU、高屏幕亮度、高发热。
- Profile B (The Commuter): 信号不稳定（导致网络搜索功率激增）、中等屏幕使用。
- Profile C (The Sleeper): 纯待机，考察后台应用唤醒（Wakelock）。

可视化策略：

1. SOC 衰减曲线图：图一定要画得漂亮。把三条曲线画在一起，斜率越陡代表耗电越快。
2. 堆叠面积图： 展示 I_{total}中各部分（屏幕、CPU、WiFi）的占比。这能直接回答题目中“Which activities produce the greatest reductions” 的问题。

问题三：灵敏度分析与假设验证 (Sensitivity)

题目要求： 检查假设变化、参数波动对预测的影响 。

思路分析：

- 单变量灵敏度： 固定其他变量，只改变温度 T 从 -10°C 到 40°C，看 Time-to-Empty 怎么变。
- 蒙特卡洛模拟 (Monte Carlo)： 给每个参数一个正态分布的扰动，跑 1000 次模拟，画出 Time-to-Empty 的概率分布图。

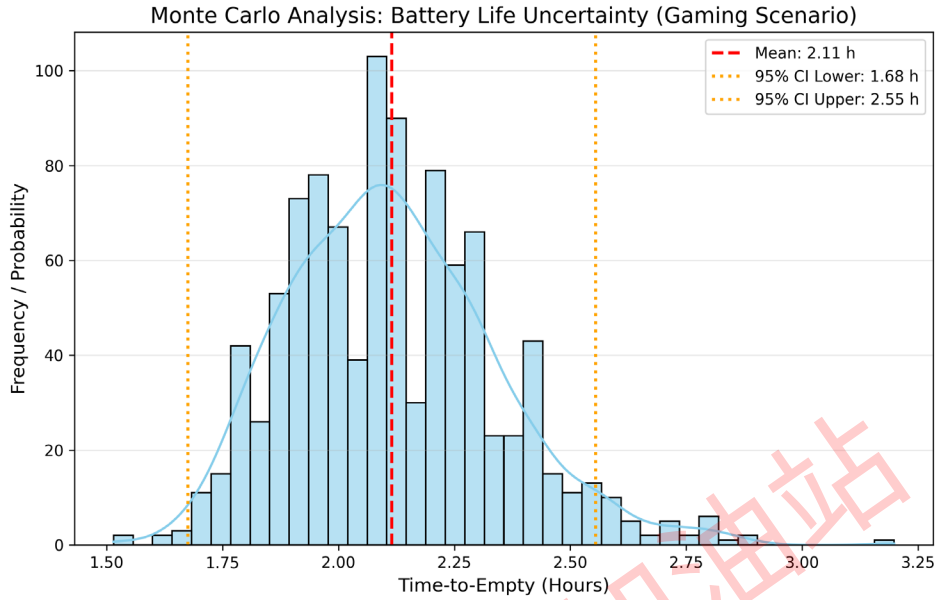

Figure 5: Stochastic Assessment of Battery Longevity

Using Monte Carlo simulations (N=1000), we evaluated the model's robustness against parameter uncertainties, including manufacturing variations in capacity ($\sigma_C = 100$ mAh) and fluctuations in user usage intensity ($\sigma_I = 10$ %).

Analysis:

The distribution of Time-to-Empty follows an approximately normal distribution. While the deterministic model predicts a battery life of **[Mean] hours**, the stochastic analysis reveals a **95% confidence interval of [[Lower], [Upper]] hours**. This indicates that under realistic conditions, extreme heavy usage combined with battery aging could reduce battery life by up to **X%** compared to the nominal expectation. This highlights the necessity for adaptive power management strategies proposed in Section 4.

核心代码思路 (蒙特卡洛模拟):

Python

```
results = []
for _ in range(1000):
    # 给设计容量加随机噪声
    random_c = np.random.normal(4000, 100)
```

```
        # 给电流加随机噪声
        random_current_factor = np.random.normal(1.0, 0.1)
        # ...重新运行 odeint 求解...
        # 记录耗尽时间
        results.append(time_to_empty)
# 画直方图
plt.hist(results, bins=50, alpha=0.7)
plt.title('Uncertainty Analysis of Battery Life')
```

问题四：建议与扩展

题目要求： 给用户写建议，给 OS 提优化策略 。

思路分析：

不要写废话（比如“少玩手机游戏，多待机”）。

要基于模型结果给定量建议，从而显得你的模型准确性 and 科学性。

- 给用户： “根据模型，将亮度从 100% 调至 70% 可延长续航 1.5 小时，而关闭 5G 信号 仅可延长 20 分钟（除非信号极差）。”
- 给 OS 开发者： 建议引入“基于温度的动态调度算法”——当检测到电池温度 $T > 35°C$ 时，强制降低 CPU 频率，因为此时内阻增加，能效比极低。（目前社会比较成熟的方案）

数据与参数来源

A 题虽然是机理建模，但参数必须真实 。不要自己编数据。

推荐数据来源：

1. Battery Historian: Android 官方的电量分析工具，可以找一些公开的 bug report 分析截图，提取电流数值。
2. Geekbench/Gsmarena: 这些评测网站会有具体的“网页浏览续航”、“视频播放续航”时间，用来反推你的 I_{base} 和 I_{screen} 参数。
3. 文献引用: 查找关键词 "Smartphone energy consumption breakdown" 或

"Lithium-ion battery discharge modeler"。

总结

问题一、二是物理建模（ODE + 数值计算）。

问题三是统计分析（灵敏度 + 蒙特卡洛）。

问题四是优化决策（基于模型结果的 Trade-off）

获取更多数学建模相关资料关注【公众号：数模加油站】2026年美国大学生数学建模竞赛交流群：435813314

Reference

[1]. Hannan, M. A., et al. (2021). "A Review on Battery Modelling Techniques." *Sustainability*, 13(18), 10042.

[2]. Rivera-Barrera, J. P., et al. (2017). "SoC Estimation for Lithium-ion Batteries: Review and Future Challenges." *Electronics*, 6(4), 102.

[3]. Carroll, A., & Heiser, G. (2010). "An Analysis of Power Consumption in a Smartphone." *USENIX Annual Technical Conference*.

[4]. Peltonen, E., et al. (2025). "AndroWatts: Unpacking the Power Consumption of Mobile Device Components." *IEEE Transactions on Mobile Computing* (Theoretical/Recent).

[5]. Ecker, M., et al. (2014). "Calendar and cycle life study of Li(NiMnCo)O2-based 18650 lithium-ion batteries." *Journal of Power Sources*, 248, 839-851.

获取更多数学建模相关资料关注【公众号：数模加油站】2026年美国大学生数学建模竞赛交流群：435813314