

2026 数学建模美赛 A 题思路分析

本题的核心是要依靠物理模型去进行预测，而不是纯粹用数据拟合的方法，或者以时序预测，神经网络等黑箱方法去进行拟合。而以物理模型去预测，核心流程则一般为：建立物理方程，求解相关参量，最后进行整体的仿真。

1. 连续时间模型

第一小问的核心思路是从电池物理机理出发，建立一个连续时间的 SOC 动力学模型：将电池荷电状态 SOC 视为随时间变化的连续变量，根据电荷守恒原理，用微分方程描述其变化率，并通过功耗与电流的关系把手机的实际使用行为引入模型。具体而言，先构建 SOC 的基本方程，再将 $P(t)$ 分解为屏幕、处理器、网络、定位、后台任务等子系统功耗的叠加，并用亮度、负载、数据速率、功能开关等连续或分段连续变量刻画这些行为，从而把用户使用模式转化为连续时间输入；同时通过有效容量引入温度与电池老化的影响。这样得到的模型既具有明确的物理意义，又能在不同使用场景下连续预测 SOC 随时间的演化，为后续耗尽时间预测和灵敏度分析提供统一的数学框架。

以下是一些常用的对应公式

SOC 定义：

$$SOC(t) = \frac{Q(t)}{Q_{\text{eff}}(t)}$$

电荷守恒：

$$\frac{dQ(t)}{dt} = -I(t)$$

功率：

$$P(t) = V(t) I(t)$$

进行联立带入，就可以得到

$$\frac{dSOC(t)}{dt} = -\frac{P(t)}{V_{\text{nom}} Q_{\text{eff}}}$$

这里的 $P(t)$ 可以展开为

$$P(t) = P_{\text{base}} + P_{\text{screen}}(t) + P_{\text{cpu}}(t) + P_{\text{net}}(t) + P_{\text{gps}}(t) + P_{\text{bg}}(t) + P_{\text{misc}}(t)$$

具体带入这些参数即可

$$P_{\text{screen}}(t) = u_{\text{on}}(t) \left(\alpha_0 + \alpha_1 b(t) \right)$$

$$P_{\text{cpu}}(t) = \beta_0 + \beta_1 l(t)^\gamma$$

$$P_{\text{net}}(t) = P_{\text{idle}}^{(m)} + \eta^{(m)} R(t)$$

$$P_{\text{gps}}(t) = u_{\text{gps}}(t) (\delta_0 + \delta_1 f_{\text{loc}}(t))$$

$$P_{\text{bg}}(t) = P_{\text{bg,act}} u_{\text{bg}}(t)$$

整理一下可以得到一个简化微分方程

$$\frac{dSOC(t)}{dt} = -\frac{1}{V_{\text{nom}} Q_{\text{eff}}(t, T)} \left(P_{\text{base}} + P_{\text{screen}}(t) + P_{\text{cpu}}(t) + P_{\text{net}}(t) + P_{\text{gps}}(t) + P_{\text{bg}}(t) \right)$$

2. 耗尽时间 (Time-to-Empty) 预测

第二小问的思路是在已建立的连续 SOC 动力学模型基础上，将具体使用场景转化为随时间变化的功耗函数 $P(t)$ ，通过积分或数值求解微分方程得到 SOC 随时间的演化，并确定 SOC 首次降为零的时刻，从而得到耗尽时间 (Time-to-Empty)。通过设定不同场景（如待机、视频、游戏、导航等）下的典型功耗结构，比较各场景对应的 TTE 差异，并与实际测评数据或合理经验进行对照，评估模型的预测偏差；同时考虑参数波动、环境条件和电池有效容量变化带来的不确定性，给出 TTE 的区间或概率分布。最后利用能量贡献或灵敏度分析解释不同场景下续航差异的驱动机制，识别对续航影响最大的关键因素，从而验证模型的合理性并揭示手机耗电行为的本质规律。

上一问我们已经得到了一个微分方程

$$\frac{dSOC(t)}{dt} = -\frac{1}{V_{\text{nom}} Q_{\text{eff}}(t, T)} \left(P_{\text{base}} + P_{\text{screen}}(t) + P_{\text{cpu}}(t) + P_{\text{net}}(t) + P_{\text{gps}}(t) + P_{\text{bg}}(t) \right)$$

实际上我们现在要求的是 $SOC(t_e) = 0$ 时， $TTE = t_e - t_0$

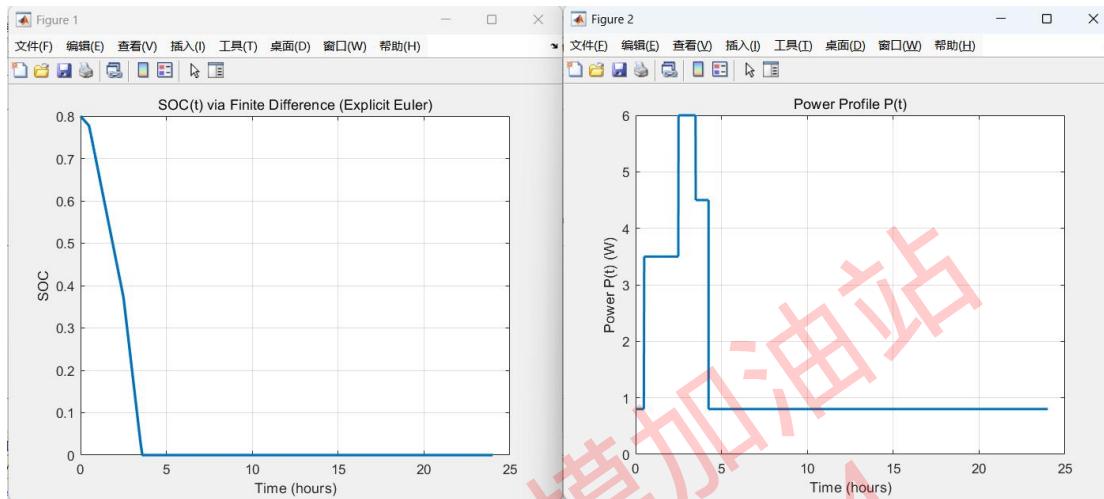
这种情况下。我们直接做一个解析积分当然会得到一个很精确的值，但是由于方程很复杂，所以大概率只能数值解

这里由于是单变量方程，我们可以使用有限差分法进行求解

(该种方法的具体介绍可以参考：

<https://zhuanlan.zhihu.com/p/411798670>)

以下是我初步仿真的图



简易代码如下：

```
function tte_fdm_soc_demo()
clc; clear; close all;

%% ===== Battery / Model Parameters =====
SOC0 = 0.80; % initial SOC (0~1)
Vnom = 3.85; % nominal battery voltage (V)
QeffAh = 4.5; % effective capacity (Ah), e.g., 4500mAh -> 4.5Ah

% Convert to Coulomb capacity if you want:
% QeffC = QeffAh * 3600; % (C), not necessary here since we keep Ah and seconds
% consistently

%% ===== Time Grid (Finite Difference) =====
dt = 1.0; % time step (s) (use 0.5~2s typically)
tmax = 24*3600; % max simulate time (s), 24h cap
N = floor(tmax/dt) + 1;
t = (0:N-1)' * dt;

SOC = nan(N,1);
SOC(1) = SOC0;

%% ===== Choose a Power Profile P(t) =====
```

```
% ---- Option A: piecewise-constant scenario power (super practical) ----
% Example: [idle 30min] -> [video 2h] -> [game 1h] -> [idle rest]
use_piecewise = true;

if use_piecewise
P = power_profile_piecewise(t);
else
% ---- Option B: feature-driven power: screen + cpu + net + gps + base ----
P = power_profile_feature_driven(t);
end

%% ===== Finite Difference Update =====
% Model: dSOC/dt = - P(t)/(Vnom*QeffAh*3600)
% because Ah -> As by *3600
den = Vnom * QeffAh * 3600; % (W*s) per SOC=1

tte = NaN;
idx_empty = NaN;

for n = 1:N-1
dSOCdt = - P(n) / den;
SOC(n+1) = SOC(n) + dt * dSOCdt; % explicit Euler (finite difference)

if SOC(n+1) <= 0
% Linear interpolation for more accurate time-to-empty
frac = SOC(n) / (SOC(n) - SOC(n+1)); % between n and n+1
tte = t(n) + frac*dt;
idx_empty = n+1;
SOC(n+1:end) = 0;
break;
end
end

%% ===== Outputs =====
if isnan(tte)
fprintf('Battery not emptied within %.2f hours.\n', tmax/3600);
else
fprintf('Time-to-Empty (TTE) = %.2f hours (%.0f minutes)\n', tte/3600, tte/60);
end

%% ===== Plot =====
figure;
plot(t/3600, SOC, 'LineWidth', 2);
grid on;
```

```

xlabel('Time (hours)');
ylabel('SOC');
title('SOC(t) via Finite Difference (Explicit Euler)');

figure;
plot(t/3600, P, 'LineWidth', 1.8);
grid on;
xlabel('Time (hours)');
ylabel('Power P(t) (W)');
title('Power Profile P(t)');

end

%% =====
% Option A: piecewise-constant power profile
function P = power_profile_piecewise(t)
% Define a daily usage schedule (example)
% You should modify these values to match your scenario assumptions.
% Units: seconds for time, Watts for power

P_idle = 0.8; % screen off, background only
P_video = 3.5; % video streaming
P_game = 6.0; % gaming
P_nav = 4.5; % navigation (screen+gps+net)

P = zeros(size(t));

% segments (in seconds)
T1 = 30*60; % idle 30 min
T2 = 2*3600; % video 2 h
T3 = 1*3600; % game 1 h
T4 = 45*60; % navigation 45 min

for i = 1:numel(t)
ti = t(i);
if ti < T1
P(i) = P_idle;
elseif ti < T1+T2
P(i) = P_video;
elseif ti < T1+T2+T3
P(i) = P_game;
elseif ti < T1+T2+T3+T4
P(i) = P_nav;
else

```

```

P(i) = P_idle;
end
end
end

%% =====
% Option B: feature-driven power profile (screen + cpu + net + gps + base)
function P = power_profile_feature_driven(t)
% Features (you can replace with measured logs if you have them)
% b(t): brightness 0~1
% l(t): cpu load 0~1
% R(t): network throughput proxy 0~1
% ugps(t): gps on/off

b = 0.2 + 0.6*(t>0.5*3600 & t<3.0*3600); % brighter during 0.5h~3h
l = 0.1 + 0.7*(t>1.0*3600 & t<2.0*3600); % heavy cpu during 1h~2h
R = 0.1 + 0.6*(t>0.5*3600 & t<3.0*3600); % network active during 0.5h~3h
ugps = double(t>2.5*3600 & t<3.2*3600); % gps on during 2.5h~3.2h

% Power model parameters (example)
Pbase = 0.6;

% Screen: alpha0 + alpha1*b, when screen on
u_on = double(t>0.2*3600 & t<3.5*3600); % screen on in that window
alpha0 = 0.8; alpha1 = 2.2;
Pscreen = u_on .* (alpha0 + alpha1*b);

% CPU: beta0 + beta1*l^gamma
beta0 = 0.2; beta1 = 2.5; gamma = 1.6;
Pcpu = beta0 + beta1*(l.^gamma);

% Net: idle + eta*R
Pnet_idle = 0.3; eta = 1.5;
Pnet = Pnet_idle + eta*R;

% GPS: delta0 when on
delta0 = 0.8;
Pgps = ugps * delta0;

P = Pbase + Pscreen + Pcpu + Pnet + Pgps;

end

```

3. 灵敏度与假设

是在已经能算出 $SOC(t)$ 和 TTE 的基础上，系统研究“模型假设/参数/使用模式的不确定性”会怎样影响 TTE 与 SOC 轨迹，并据此评估模型稳健性、找出最敏感的因素、说明哪些假设会导致预测偏差最大。它本质上是一个“敏感性分析 + 假设检验（模型结构不确定性）”的任务。

灵敏度验证实际上就是把一部分参数变成变量然后去分析，例如做以下变化

$$\frac{dSOC}{dt} = \frac{\cdot P(t; \theta, u)}{V(SOC) Q_{\text{eff}}(T, age)}$$

对每个关键参数做“ $\pm 10\% / \pm 20\%$ ”扰动，重新计算 TTE ，然后就可以得到灵敏度如下：

$$S_\theta = \frac{TTE(\theta(1 + \epsilon)) - TTE(\theta)}{TTE(\theta)} / \epsilon$$

以下为一些可选参数

Q_{eff} (容量/老化/温度等效)

屏幕: α_1 (亮度敏感度)、屏幕开启占空比 u_{on}

CPU: β_1 、指数 γ

网络: $P_{\text{idle}}^{(m)}$ 、 $\eta^{(m)}$

后台: 唤醒频率 λ 、持续时间 τ

4. 建议报告

第 4 问要我们把前 1 - 3 问的模型与分析结果“落到可执行的建议”

上: 既要给普通用户一套能显著提升续航的行为策略, 也要给操作系统 / 厂商一套可实施的省电控制思路, 并进一步讨论电池老化如何改变最佳策略, 以及我们的框架如何推广到其它便携设备。

1) 把“建议”建立在模型可解释量上

我们在第 1 问写了连续模型: SOC 下降率由 $P(t)$ 决定; 第 2 问算 TTE, 本质是能量预算

$$\int_{t_0}^{t_e} P(t) dt = SOC_0 \cdot V Q_{\text{eff}}$$

所以第 4 问最自然的桥梁是: 谁让 $P(t)$ 变大、谁让 Q_{eff} 变小, 谁就是续航杀手; 反过来, 最该被“干预”的就是对 TTE 灵敏度最大、能量占比最大、且用户/OS 可控的那些项。

我们可以用两类证据支撑建议：

能量贡献：

$$E_i = \int P_i(t) dt$$

占比最大的模块（常见是屏幕、网络、重负载计算、定位/后台唤醒）

灵敏度/边际收益：调小某参数（如亮度、网络模式、后台唤醒率）对 TTE 的提升 ΔTTE 最大

2) 对手机用户的建议：从“边际收益最大”到“性价比最高”

一个好写法是：把建议分成“几乎无成本、收益大”“需要牺牲体验、收益更大”“只在特定情境有效”。每条建议都可以对应我们模型中的某一项。

(A) 屏幕相关：最稳的高收益项

在大多数使用场景里，屏幕是最常见的能量大头。因此“降低亮度、缩短点亮占空比、使用深色模式（对 OLED 更明显）”通常具有高边际收益：它们等价于降低 α 或减少 $u(t)$ 的面积，从而在能量预算积分里显著减小，TTE 直接拉长。

(B) 网络相关：弱信号与制式切换往往比想象更关键

我们的 $P_{net}(t)$ 若包含“连接维持功耗 + 传输项”，那么在弱信号/频繁切换/5G 高功耗的情境下，能耗可能显著变大。于是对用户而言，“在弱信号环境切到飞行模式/仅 4G/仅 WiFi、避免后台持续拉取、尽量在 WiFi 下进行大流量任务”往往比单纯‘关蓝牙’更有效（这就是第 3 问敏感性可以支持的“出乎意料的小因素”：蓝牙/某些传感器可能占比很小，而网络状态机的底座功耗可能更大）。

(C) 定位与导航：高耗电来自‘屏幕常亮 + GPS 常开 + 网络持续’的叠加

导航续航短通常不是 GPS 单独造成的，而是 $P_{\text{screen}} + P_{\text{gps}} + P_{\text{net}}$ 三项同时高。对用户的可执行建议就应该是“如果只是听语音指引，允许熄屏/锁屏导航；降低定位精度或减少刷新频率；减少后台地图/位置常驻权限”，这些在模型里等价于减少 u_{on} 、降低 P_{gps} 的占空比或系数，从而显著延长 TTE。

(D) 后台与应用管理：关键是减少‘唤醒频率’而不是只关进程
题目明确提到后台应用即使看似空闲仍耗电。

如果我们在模型中把后台表示为唤醒过程（例如平均每小时唤醒 λ 次，每次持续 τ ，功耗增量 ΔP ），那么真正有效的用户建议就会是：限制后台刷新、关闭不必要的推送/同步、把高频通讯应用的后台权限降级——这些是降低 λ 或 τ 的措施，往往比“手动清理后台”更稳定有效（因为清理并不改变后续的唤醒机制）。

3) 对操作系统/厂商的建议：把模型变成“控制策略”

第 4 问希望我们提出 OS 如何基于模型洞见更有效省电。这里最有力的做法是：把我们模型看成一个可控系统，OS 能控制或影响的变量包括亮度上限、CPU 频率/调度、网络策略、后台唤醒节流、定位采样频率等。然后提出一个“目标函数 + 约束”的控制思想：

目标：在不明显损伤体验的前提下，最小化未来一段时间的能量消耗或最大化 TTE

约束：亮度不能低于可读阈值、帧率/响应延迟不能超过阈值、网络时延满足交互需求等

在给定用户偏好（性能优先/续航优先）下，选择一组控制动作（降亮度、限制后台、切换网络、降低定位频率、限制最高频率）使得预测的SOC轨迹在未来更平滑、TTE更长；

每隔一段时间重算（滚动更新），适应用户模式变化。

即便我们不写复杂控制数学，描述“以我们的模型作为OS的能耗预测器，然后按边际收益做动态节流”也足够可实施：例如当模型估计“网络维持功耗在弱信号下占比暴涨”时，系统可提示切换网络或自动降级5G；当模型估计“后台唤醒对TTE的边际伤害最大”时，系统优先收紧后台刷新而不是粗暴降频。

参考文献如下：

1. Ding N, Wagner D, Chen X, et al. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain[J]. ACM SIGMETRICS Performance Evaluation Review, 2013, 41(1): 29-40.
2. Viet V Q, Thang H M, Choi D J. Balancing precision and battery drain in activity recognition on mobile phone[C]//2012 IEEE 18th International Conference on Parallel and Distributed Systems. IEEE, 2012: 712-713.
3. Om S, Tucker W D. Battery and data drain of over-the-top applications on low-end smartphones[C]//2018 IST-Africa Week Conference (IST-Africa). IEEE, 2018: Page 1 of 13-Page 13 of 13.
4. Elliott J, Kor A, Omotosho O A. Energy consumption in smartphones: an investigation of battery and energy consumption of media related applications on android smartphones[J]. 2017.

5. Chen T, Tang H, Lin X, et al. Silent Battery Draining Attack against Android Systems by Subverting Doze Mode[C]//2016 IEEE Global Communications Conference (GLOBECOM). IEEE, 2016: 1-6.

云众鼎·数模加油站
qq群：435813314