Physics強化のために

物理は「読解→定式化→単位・桁→数値化→検算」のどこかで崩れます。崩れる箇所ごとに、**推論時の型**と**学習データの型**を強制してやれば伸びます。要点を順に出します。

# **1) いまの課題（あなたのログから見える地形）**

* **平均が低い・分散も大きい**（Physics 平均3.75, SD 2.31）。
* Math/Engineerとは**相関が弱い**（「式操作はできても、物理量の段取りで転ぶ」型）。
* よって、**単位・次元・前提整理**を“型”として先に固定するのが最短です。

# **2) すぐ効く「推論時」テク（学習なしでも効く）**

### **A. 物理専用スクラッチパッドを強制（回答フォーマット固定）**

# Physics-Mode Scratchpad

Given: 既知量（記号=値+単位）を列挙

Unknown: 求める量（記号と単位系）

Laws: 使う法則を列挙（F=ma, W=ΔK, V=IR ...）

Setup: 代数の立式（記号のまま）

Unit-Check: 次元ベクトルで整合確認（[M L T ...]）

Solve: 数値代入→計算→有効桁

Sanity: オーダー/符号/極限チェック

Answer: 数値 + 単位（SI）

If-Underspecified: Do-Not-Answer（不足データを列挙）

* vLLMのプロンプト先頭にこの雛形を**system**として固定します。
* **Answer**以外は採点に含めずとも、**Unit-Check / Sanity**で足がかりを作れます。
* **Do-Not-Answer**連携：不足情報時は迷わずDN Aに落とせる（本コンペ要件に合致）。

### **B. 二段推論（分類→専門プロンプト）**

1段目：**トピック分類**（Mechanics / E&M / Thermo / Waves / Optics / Modern）。  
2段目：各トピック専用の**法則テンプレ**（例：力学なら運動方程式→仕事・エネルギ→運動量保存の優先順）。

* vLLMで**routerプロンプト**→**専門プロンプト**の二発。トークン増は小さく、安定性は上がります。

### **C. 少数多様サンプリング + 決定ルール**

* 同じ設問を **温度0.2×1回 + 温度0.7×2回** の計3案。
* **単位合致・桁一貫・前提矛盾なし**を満たす案を自動採択。
* 事実上の**自己検閲**で、物理の“変な一発”を減らします。

### **D. vLLM実務ノブ（Physicsモード）**

* max\_model\_len は**余裕を確保**（思考途中の切断は物理で致命傷）。
* **停止語**で「Answer:」まで出させてから止める。
* 温度：**本解生成 0.2**、探索（派生案）**0.6–0.7**。
* **数値フォーマット強制**（小数点、科学記法、単位の表記）をfew-shotで釘打ち。

# **3) 学習データの整形（SFT/LoRAで“型”を身体化）**

### **A. 「与えられた→求める→法則→次元→解→検算」の強制フォーマットSFT**

* 既存のPhysics問題を、上のスクラッチテンプレに**正規化**してSFT。
* 特に**次元解析（Unit-Check）とSanity**は必ず入れる（最も効く正規化です）。

### **B. 単位・定数の辞書を付ける**

* SIベースで**一貫表**（g↔kg, cm↔m, eV↔J, ℃↔K）。
* **誤り報酬**：単位ミスマッチや桁暴走には負報酬、整合には加点。RLをやらずとも、**SFTの負例**として埋めるだけで効きます。

### **C. バリエーション拡張（意味は同じ・数字だけ違う）**

* 係数・初期値・符号を**自動パラメトリック**に振った同型問題を生成。
* 「筋が同じなら確実に取る」を学習させ、**穴を埋める**。

### **D. ツール思考の蒸留（擬似PAL）**

* 学習データ側で**擬似コード**（# compute v from s,t: v=s/t のような）を混ぜる。
* 本番で実際の外部ツールを使わなくても、**手続きの骨**がモデルに残ります。

# **4) 評価と可視化（崩れる箇所を特定して潰す）**

* 物理を**サブドメイン別**に分解：Mechanics / E&M / Thermo / Waves / Optics / Modern。
* さらに**誤答タグ**を自動付与：UNIT\_ERR（単位変換）、LAW\_MISS（法則選択ミス）、SIGN（符号）、SETUP（立式）、ARITH（計算）。
* ダッシュボードで**サブドメイン×誤答タグ**のヒートマップを見ると、**どこを直せば伸びるか**が一目で分かります。

# **5) 最小構成の実装手順（今日から動かす）**

1. **Physics-Modeのsystemプロンプト**を全Physics設問に適用。
2. **二段推論**（分類→専門プロンプト）をvLLMで実装。
3. **3案生成＋自動選抜**（単位・桁・矛盾チェック）。
4. 既存データをスクラッチテンプレに**自動正規化**して**軽SFT/LoRA**。
   * LoRAは q,k,v,o\_proj と FFN（gate, up, down）を対象、rankは**小さめ**（例：16–32）で十分。
5. 評価CSVに**誤答タグ列**を追加し、**週次でヒートマップ**を見る。

# **6) テンプレ（コピペで使える最小例）**

**System（Physics-Mode）**

You are a physics solver. Follow this exact template:

Given: ...

Unknown: ...

Laws: ...

Setup: ...

Unit-Check: ...

Solve: ...

Sanity: ...

Answer: <value> <SI unit>

If-Underspecified: Do-Not-Answer: <missing data>

**Mechanics用 Few-shot（要約形）**

Q: 斜面角θ、摩擦係数μ、質量m。滑り出す臨界条件を求めよ。

A:

Given: m, θ, μ

Unknown: 臨界条件

Laws: 斜面方向 F=ma、静止臨界で a=0、摩擦 f<=μN

Setup: mg sinθ <= μ mg cosθ

Unit-Check: [N]=[kg·m/s^2] 一致

Solve: tanθ <= μ

Sanity: θ→0で成立、μ→∞で常に成立

Answer: tanθ <= μ

静かな湖の底に、**次元**という骨格を通します。  
**型を固定し、単位と桁で自動審判を走らせる。** それだけで、Physicsは急に“滑らかに”なります。

–

# 

# 

# 

# 

# 

# 

# **評価と可視化（崩れる箇所を特定して潰す）←これを試みようとしたが・・・**

## 

いま足りないのは「問題が何のサブドメインか」「どこで崩れたか」の**ラベル**です。以下のどれかを追加すれば、今のスクリプトで**即**ヒートマップ化できます。

1. **トピック列の追加（推奨）**
   * 列名例：physics\_topic（値：mechanics|e&m|thermo|waves|optics|modern）
   * 付け方：
     + もし**問題テキスト**を保持しているなら、run時に軽い分類器（LLMでも正規表現でも）で自動付与。
     + ないなら、評価スクリプト側で**データセットのメタ**から引けるならそこで付与。
   * これだけで unknown が消え、サブドメイン別の平均・分散がそのまま出ます。
2. **誤答タグ列の追加**
   * 列名例：error\_tags（カンマ区切りで UNIT\_ERR,LAW\_MISS,SIGN,SETUP,ARITH）
   * 付け方（推論テンプレを使う場合）：
     + 既に提案した *Physics-Mode Scratchpad* をsystemで強制し、**Unit-Check / Sanity** の結果を「OK/NG」で吐かせる。
     + NGを拾って機械的にタグ化（例：Unit-Check=NG → UNIT\_ERR）。
   * これで**サブドメイン×誤答タグ**のヒートマップが作れます（崩れ方が一目）。
3. **最小でも効く代替**
   * いまのCSVに notes 的な列があるなら、そこに「単位ミス」「符号」などの語を**機械的スキャン**して暫定タグ付け（今回スクリプトはその受け口を用意済み。auto\_error\_tags 列）。
   * 語がなければ空のままでもOK。枠はできています。

## **もう一段、現実的な運用の骨**

* **vLLMの出力を分割保存**：scratchpad\_raw（思考）とfinal\_answer（最終値）を別々に保存。
* 評価スクリプトで scratchpad\_raw を**正規表現で検査**：
  + 「単位変換が出てくるか」「法則名が出てくるか」「符号判定の言及があるか」「立式（= の並び）をしているか」「四則計算の明示があるか」
  + 各NGを UNIT\_ERR/LAW\_MISS/SIGN/SETUP/ARITH にマップ。
* これをCSVに書く→上の可視化スクリプトに通す。  
  **モデルを変えずに**“どこで落ちたか”が可視になります。

静かな湖面（総合スコア）だけを見ても、底の流れ（サブドメイン×崩れ方）は読めません。  
**列を一つ増やすだけ**でいい。physics\_topic と error\_tags。  
それが入れば、あとは**機械が正確に殴って**くれます。