有効加速チェッカー

Valid Accel Checker

ウマ娘プリティダービー有志ツール — 設計仕様書 v4.1

2026年2月

1. システム概要

1.1 ツール概要

本ツールは、ウマ娘プリティダービーのレース条件と育成ウマ娘の特性を入力することで、該当レースで発動可能な加速スキルを一覧表示し、各スキルの発動タイミングを「最速 / 準速 / ランダム有効 / 無効」に分類するWebアプリケーションである。

ユーザーが距離・バ場・バ場状態・開催場所などのレース条件を入力すると、データベース内のスキルをフィルタリングし、「タイミングの早さ」と「不確実性の低さ」を軸にソートした結果を一覧で返す。

1.2 バージョン履歴と v4.1 の変更点

バージョン	主な変更点
v1.0	初版。基本フィルタ・タイミング判定・最速/準速/無効の3段階判定を実装。
v2.0	OR条件（segs配列）対応、スキルデータの大幅拡充、坂判定ロジック追加。
v3.0	内部条件の3区分分類（固定/レース運び/ランダム）と不確実性スコア（unc）の導入。ソート基準にuncを追加。
v3.2	固有スキル・進化スキルについてキャラクター名を併記
v3.3	① 不確実性スコアの重み改訂（rr: +2、rv: +4）。② 新カテゴリ「レース運び条件(弱)」の追加（+1）。③ hp_per / order / order_rate / popularity を固定条件へ再分類（±0）。④ uncをデータ埋め込みからランタイム動的計算（calcUnc関数）へ移行。
v4.1	【本書対象】① スキルデータ・レースプリセット・キャラクターデータを外部JSファイルに分離。② データ読み込み方式をasync fetch()から同期的な<script src>タグへ移行。③ モジュール構成によりGitHub Pages等の静的ホスティング環境への対応を改善。ゲームロジックの変更なし。

2. 設計概要

2.1 設計思想

v3.3 以降の核心は「スキル発動条件をその性質によって3つに分類し、再現性の低い条件ほど高い不確実性スコアを付与する」というアプローチである。

発動タイミングの早さだけでなく、そのタイミングが毎レース安定して訪れるかどうかを同時に評価することで、実戦での運用優先度をより正確に判断できるようにすることを目指す。

v4.1 ではこのロジックに変更を加えず、ファイル構成のみをモジュール化した。

2.2 内部条件の3区分

スキルデータの raw フィールドに記録された内部条件を、以下の3カテゴリに分類する（v3.3 より変更なし）。

① 固定条件（Fixed Conditions）

レースが決まった時点で確定し、毎レース同一の結果をもたらす条件であり不確実性への寄与はない。

変数例	概要
distance_type	距離区分（短距離/マイル/中距離/長距離）
ground_type	バ場種別（芝/ダート）
ground_condition	バ場状態（良/稍重/重/不良）
running_style	作戦（逃げ/先行/差し/追込）
phase	レース区間（序盤/中盤/終盤）
course_distance	具体的なレース距離（m）
track_id	レース場ID
is_finalcorner, is_last_straight など	コース上の地点フラグ（最終コーナー、最終直線など）
activate_count_start, activate_count_all_team など	スキル発動数カウント

※v3.3 にて以下の変数が新たに固定条件へ再分類（v4.1 で変更なし）。

変数例	概要	分類変更前（v3.2）
hp_per / hp_per<=30	残り持久力（%）	レース運び条件 (+2)
order	具体的な順位	レース運び条件 (+2)
order_rate / order_rate>50 / order_rate>40	順位の割合（%）	レース運び条件 (+2)
order_rate_in[X]_continue / order_rate_out[X]_continue	特定の順位率の継続維持	レース運び条件 (+2)
popularity	人気順	レース運び条件 (+2)

② レース運び条件(弱)（Weak Race-Running Conditions / rr配列）【v3.3 新設】

rr 配列に格納されるが、他のレース運び条件より不確実性への影響が小さい変数。1変数につき +1 の不確実性スコアが加算される。UIでは黄色タグで表示される。

変数例	概要
infront_near_lane_time	前方近くのレーンに他ウマ娘がいる時間
behind_near_lane_time	後方近くのレーンに他ウマ娘がいる時間

③ レース運び条件（Race-Running Conditions / rr配列）

レース展開の中でリアルタイムに変化する条件。スキルデータの rr 配列に格納される。1変数につき +2 の不確実性スコアが加算される（v3.2 では +1 から重み増）。

変数例	概要
is_overtake	現在追い抜きを試みているか
change_order_onetime	直近で着順が変化したか
blocked_front	前方が塞がれているか
blocked_front_continuetime	前が塞がれている継続時間
blocked_side_continuetime	横が塞がれている継続時間
overtake_target_time	詰め寄られている時間
bashin_diff_infront, bashin_diff_behind	前後との距離差（バ身単位）
distance_diff_top, distance_diff_rate	先頭との距離差
is_move_lane	コース変更中か（左/右）
temptation_count	掛かり回数
compete_fight_count	追い比べ回数
activate_count_heal	スキルによる回復発動回数
activate_count_all_team	チーム全体のスキル発動数
is_used_skill_id	特定スキルが既に発動済みか
is_behind_in	外側から抜いているか

④ ランダム条件（Random Conditions / rv配列）

レースごとに不確定な要素によって決まる条件。スキルデータの rv 配列に格納される。1変数につき +4 の不確実性スコアが加算される（v3.2 では +3 から重み増）。v3.3 では popularity が固定条件へ再分類。

変数	概要	不確実な理由
post_number	枠番（1〜18）	抽選で決定されるため事前に確定しない
motivation	やる気（1〜5、5=絶好調）	育成・レース進行によって変化

2.3 不確実性スコア（unc）の計算式

スキルごとの不確実性スコアは、v3.3 からランタイムの calcUnc(s) 関数で動的計算される（v4.1 で変更なし）。

unc = (rr変数のうちレース運び条件(弱)の数 × 1) + (rr変数のうちレース運び条件の数 × 2) + (rv変数のうちランダム条件の数 × 4)

スキル例	rrの変数	rvの変数	unc計算	unc値
下り坂系スキル（坂条件のみ）	なし	なし	0	0
逃げ系スキル（order付き）	order → 固定条件	なし	0	0
垂れウマ回避（hp_per+infront）	hp_per, infront_near_lane_time	なし	1×1 + 0×2 + 0×4	1
コンドル猛撃波（order+is_overtake）	is_overtake	なし	0×1 + 1×2 + 0×4	2
セイリオス（order_rate+popularity）	distance_diff_top	popularity → 固定条件	0×1 + 1×2 + 0×4	2
インサイドヒーロー（post_number）	なし	post_number	0×1 + 0×2 + 1×4	4

2.4 UIでの不確実性表示

不確実性スコアは以下の5段階でカードに表示される（v3.3 より変更なし）。

unc値	ラベル	意味
0	確定	毎レース必ず同じ条件で発動。最も信頼性が高い。
1-2	低	軽微なレース運び依存が1〜2変数。おおむねコントロール可能。
3-4	中	複数のレース運び変数が絡む。運用上は概ね安定。
5-6	高	多数の変数またはランダム要素が関与。発動率が安定しない場合がある。
7以上	最高	ランダム変数を含む複合条件。対象スキルは発動機会が限られる。

タグ色	条件区分	不確実性への寄与
黄色タグ	レース運び条件(弱)（rr）	+1 / 変数
橙色タグ	レース運び条件（rr）	+2 / 変数
赤色タグ	ランダム条件（rv）	+4 / 変数

3. 発動タイミング判定ロジック

本章の内容は v3.2 から変更なし（v4.1 でも同様）。参照のため再掲する。

3.1 条件タイプの分類

内部条件の「固定条件」を解析し、スキルの発動タイミングを算出する。

tm.type 値	発動タイミング方式
fixed	fixed_point に基づき、レース距離とコース情報から発動地点を計算する
random (phase系)	レース終盤・中盤後半などのランダム発動を表すラベルを付与する
slope_down / slope_up	入力された坂情報と照合し、有効か無効かを判定する
slope_up_later_half	後半の上り坂区間との一致で判定する

3.2 fixed_point ごとの計算ロジック

fixed_point	発動地点の計算方法
ls（ラストスパート）	入力値 manLS、またはプリセット ls、またはデフォルト distance×2/3 を使用。発動地点 = ls
finalcorner（最終コーナー）	入力値 manFC、またはプリセット fc を使用。発動地点 = fc
finalstraight（最終直線）	入力値 manFS、またはプリセット fs を使用。発動地点 = fs
remain_lte（残りX m以下）	distance - rv = 発動地点（固定点として扱える場合は最速判定）
remain_range（残りX m前後）	distance - rv = 発動地点として計算
dist_rate_after（距離rate%以降）	distance × rv / 100 = 発動地点
phase2_corner（終盤コーナー）	最終コーナーまたはコース依存でランダム判定
phase2_front_straight（向こう正面）	終盤前の向こう正面通過タイミングを推定
phase2_corner_nonfinal（終盤の非最終コーナー）	コース依存として扱い、ランダム判定ラベルを付与

3.3 最速・準速・無効の判定基準

fixed_point が確定した場合、judgeOffset 関数により以下の基準で分類される。

判定結果	cls（内部）	条件
最速発動	b-fast	オフセット offset が -50m ≤ offset ≤ 0m の範囲
準速発動	b-semi	オフセットが 0m < offset ≤ 100m の範囲
無効（早期）	b-inv	条件地点がラストスパートよりも 50m 以上前になる場合
無効（遅延）	b-inv	条件地点がラストスパートよりも 100m 以上後になる場合

offset = 発動地点 **−** **ls**（ラストスパート開始地点）

3.4 タイミング優先値（pri）一覧

ソートの第1キーとして使用する。数値が小さいほど優先順位が高い（早い）タイミングである。

pri値	タイミング種別
0	最速発動（fixed, ls基準で-50〜0m）またはランダム類（remain_lte で最速判定された場合）
1	準速発動（fixed, ls基準で0〜+100m）
2	常時有効スキル
10	上り坂（有効）
11	下り坂（有効）
20	ランダム有効（終盤前1/4 = fq）
21	ランダム有効（終盤前半 = fh）
22	ランダム有効（終盤全般 = phase2）
23	ランダム有効（最終コーナー = finalcorner_random）
24〜29	その他ランダム有効（lh, straight, all_corner, after50 等）
35〜36	ランダム有効（その他）
40〜41	坂情報未入力のため判定保留
50	コース情報不足（fc/fs 未入力）
90〜99	無効・判定不能

4. 検索・ソート処理（doSearch）

4.1 入力パラメータ

パラメータ	型	説明
dist	整数	レース距離（m）。選択リストから選択
gt	整数	バ場種別（1=芝, 2=ダート）
gc	整数	バ場状態（1=良, 2=稍重, 3=重, 4=不良）
mv	整数|null	やる気（1〜5、null=指定なし）
venue	文字列	開催場所名（レース場グループ）
manLS	浮動小数|NaN	ラストスパート開始地点（m）。空欄時はプリセット or デフォルト
manFC	浮動小数|NaN	最終コーナー開始地点（m）
manFS	浮動小数|NaN	最終直線開始地点（m）
upS / dnS	配列	上り坂・下り坂の区間リスト（{start, end}）

4.2 フィルタリング処理（matches 関数）

以下の条件をすべて満たすスキルのみが検索対象となる。

- 距離区分（dt）が一致するか、dt配列が空（全距離対応）
- バ場種別（gt）が一致するか、gt配列が空
- バ場状態（gc）が一致するか、gc配列が空
- やる気条件（mv）を満たすか、mv条件がない
- レース場（tids）が一致するか、track配列が空
- 距離範囲条件（cd_ge, cd_le, cd_eq）を満たす
- レア度フィルターを満たす（rarFilter による絞り込み）
- 全セグメントが exclude タイプでない

4.3 ソート基準

フィルタリング後の結果を以下の優先順序でソートする（v3.3 より変更なし）。

優先順	キー	方向	説明
1位	res.pri（タイミング）	昇順	発動タイミングが早いスキルを上位に
2位	unc（不確実性）	昇順	同タイミング内で、不確実性が低いスキルを上位に。v3.3 からランタイム動的計算
3位	qty（効果量）	降順	同不確実性内で、効果量が大きいスキルを上位に
4位	n（スキル名）	日本語50音順	最終的な同一条件内の順序

5. スキルデータ仕様（SKILLSオブジェクト）

5.1 スキルオブジェクトのフィールド定義

フィールド名	型	説明
n	文字列	スキル名
e	文字列	スキル効果テキスト
dt	整数[]	距離区分（1=短, 2=マイル, 3=中, 4=長）。空配列=全距離対応
gt	整数[]	バ場種別（1=芝, 2=ダート）。空=全バ場
gc	整数[]	バ場状態（1〜4）。空=全バ場状態
mv	整数|null	やる気要件最小値（null=不問）
track	整数[]	対象レース場ID。空=全レース場対応
cd_ge / cd_le / cd_eq	整数|null	距離条件（以上/以下/等しい）
tm	オブジェクト	発動タイミング情報（type, fixed_point, subtype, rv）
segs	配列	OR条件の各セグメント。calcResult はこの配列から最良priを選択
raw	文字列	ゲーム内部の生条件テキスト（デバッグ用）
qty	浮動小数	効果量（÷10000 した値が表示値）
dur	整数	効果時間（ミリ秒）。-1=永続
rar	整数	レア度内部値（1=白, 2=金, 3=固有下位, 4=固有, 5=固有, 6=進化）
rarj	文字列	レア度表示名
rr	文字列[]	レース運び条件の変数名リスト
rv	文字列[]	ランダム条件の変数名リスト
unc	整数	不確実性スコアデータ埋め込み値。v3.3以降はランタイムの calcUnc(s) を優先使用

5.2 外部データファイル仕様（v4.1 新設）

v4.1 では、従来HTMLに埋め込まれていたデータを以下の3つの外部JSファイルに分離した。

ファイル	変数名	件数目安	内容
data/skills.js	SKILLS	〜350件	全スキルオブジェクトの配列
data/race_presets.js	RACES	51件	レースプリセット（ls/fc/fs/坂情報）の配列
data/skill_chara.js	SKILL_CHARA	〜876件	スキルIDとキャラクター名のマッピング

データファイルのフォーマット

各ファイルはブラウザのグローバルスコープで参照できるよう、var 宣言を使用する（const はスクリプトスコープに閉じるため不可）。

```
// 例: data/skills.js
```

var SKILLS = [{ n: "スキル名", e: "効果", ... }, ...];

// 例: data/race_presets.js
var RACES = [{ name: "東京2400", dist: 2400, ls: 1600, fc: 1450, fs: 1700, ... }, ...];

// 例: data/skill_chara.js
var SKILL_CHARA = { "100001": "スペシャルウィーク", ... };

6. ファイル構成とデータ読み込み（v4.1 新設）

6.1 ディレクトリ構成

v4.1 のファイル構成は以下の通り。

パス	説明
index.html	メインアプリケーションファイル（HTML/CSS/JS）
data/skills.js	スキルデータ（var SKILLS）
data/race_presets.js	レースプリセットデータ（var RACES）
data/skill_chara.js	キャラクター名マッピング（var SKILL_CHARA）

6.2 データ読み込み方式の変更（v3.x → v4.1）

v3.x では単一のHTMLファイルによる実装を採用していたが、v4.1 では同期的な <script src> タグに変更した。

項目	v3.x（旧）	v4.1（新）
データ形式	単一のHTMLファイル	data/*.js（JSファイル、var宣言）
読み込み方式	–	<script src="..."> タグ（同期）
初期化関数	async initApp()	DOMContentLoaded イベント
グローバル変数	const SKILLS = [] HTML上で宣言	var SKILLS = [...] として直接定義
エラー要因	–	なし（スクリプトロード失敗のみ）
静的ホスティング適性	基本どんな環境でも動作可能	file:// プロトコルでも動作可能

6.3 HTMLでのスクリプト読み込み順序

index.html の </body> 直前に以下の順序でスクリプトタグを配置する。データファイルを先に読み込むことで、メインスクリプトからグローバル変数を参照できる。

```
<script src="data/skills.js"></script>
<script src="data/race_presets.js"></script>
<script src="data/skill_chara.js"></script>
<script>  /* メインロジック（SKILLS / RACES / SKILL_CHARA を参照） */  </script>
```

6.4 変数スコープに関する注意

const はスクリプトスコープに閉じるため、異なる <script> タグ間で参照できない。データファイルでは必ず var を使用すること。

宣言方式	スコープ	スクリプト間参照	採用
const SKILLS = [...]	スクリプトスコープ	不可	× 不採用
let SKILLS = [...]	スクリプトスコープ	不可	× 不採用

宣言方式	スコープ	スクリプト間参照	採用
var SKILLS = [...]	グローバルスコープ（window）	可	✓ 採用

7. フィルタ・UI仕様

7.1 レア度フィルター

表示ラベル	rarFilter値	対象rar
白	1	rar=1（白スキル）
金	2	rar=2（金スキル）
固有	4	rar=4, 5（固有スキル・どちらも同じカテゴリとして正規化）
進化	6	rar=6（進化スキル）

7.2 コース情報入力

- レースプリセット：51種のレース（プリセットSel）から選択すると ls/fc/fs/坂情報が自動入力される
- 手動入力：manLS, manFC, manFS フィールドで個別上書きが可能
- 坂区間：「+坂を追加」ボタンで上り坂・下り坂の開始〜終了地点を複数登録可能
- 手動入力値がある場合、プリセット値より優先される

7.3 着順・位置率の推定表示

internal条件に含まれる order_rate や order から、9頭立て・12頭立てでの着順目安をカード内に表示する。

7.4 警告表示

fc（最終コーナー）や fs（最終直線）が必要なスキルが検索結果に含まれているにもかかわらず、それらの値が未入力の場合、警告ボックスを表示する。

8. 設計に関する Q&A

Q1. v4.1 でなぜ単一HTMLから <script src> に変更したのか？

計算のし直しの度にHTMLファイルを直接触らなきゃいけなくてしんどかったし、ミスも増えていたため。

<script src> タグによる読み込みはブラウザの標準的なスクリプトロード機構を使用するため、静的ホスティング（GitHub Pages 等）やローカルファイルとして開いた場合でも動作する。

Q2. v4.1 でゲームロジックは変わっているか？

変わっていない。v4.1 はデータファイルの外部化とローディング方式の変更のみであり、スキルフィルタリング・タイミング判定・不確実性スコア計算・ソートロジックは一部の不具合を除きすべて v3.3 から継承されている。

Q3. なぜ v3.3 でレース運び条件の重みを +1 → +2 に変更したのか？

v3.2 では rr 変数と rv 変数の重み比が 1:3 だったが、実運用フィードバックから「レース運び条件がより強く不確実性に影響する」との評価を受けた。重みを rr: +2・rv: +4 へ引き上げ（比率は同じ 1:2）、スコア全体のレンジを広げることで分解能を向上させた。

Q4.「レース運び条件(弱)」はなぜ別カテゴリとして設けたのか？

短距離・マイルの影響力が強い「ノンストップガール」系統のスキル順位の実質的な上方修正。
infront_near_lane_time / behind_near_lane_time は「近くのレーンにウマ娘がいる時間」という緩やかな位置関係を表す変数であり、他のレース運び条件（ブロック状態・バ身差等）と比べて発動への影響度が低いと判断された。これを +2 と同等に扱うと過大評価になるため、+1 の独立カテゴリとして分離した。

Q5. hp_per / order / order_rate / popularity はなぜ固定条件に再分類されたのか？

popularityについては日本ダービーにおける「セイリオス」の実質的な順位上方修正。また、順位条件についてもアナボ・アンスキなど同様に順位上方修正。これらの変数はレース中に変化するものの、スキルの発動条件として設定された数値は「育成方針・作戦・人気設定」によってほぼ事前に見通せると再評価された。発動の不確実性への寄与が実質的に小さいことから、固定条件（±0）へ移行した。

Q6. unc のランタイム動的計算への移行はどのような影響があるか？

SKILLSデータの unc フィールドは引き続き存在するが、v3.3 からは calcUnc(s) 関数の返り値が優先使用される。
FIXED_RR_VARS・FIXED_RV_VARS セットを編集するだけで変数の分類変更が即時反映されるため、将来の調整コストを低減できる。

Q7. OR条件（segs）がある場合の処理はどうなるか？

calcResult 関数が segs 配列の各セグメントを calcSegTiming で評価し、最も pri 値が小さい（早い）結果を採用する。どのOR条件が一番早く発動しうるかを自動的に選択する。

9. 既知の制限事項

- スキルデータはハードコードされており、新スキル追加にはデータ生成スクリプトによる更新が必要。
- 不確実性スコアは「変数の数と種別」に基づく定量評価であり、各変数の実発動頻度やゲームシミュレーション結果は反映されていない。
- 発動タイミングのオフセット計算はラストスパート基準。ラストスパートを用いないスキル（中盤発動等）の順序精度は制限される（地固めなど）。
- ORDER_RANK表示（着順目安）は9頭立て・12頭立てを基準としたおおまかな目安であり、実際のレース頭数に依存する。
- 坂判定は手動入力またはプリセットデータに依存するため、プリセット未登録のコースでは精度が低下する可能性がある。
- 天候条件（weather）は現在のフィルタに含まれていない。天候依存スキル（晴/雨）は手動で確認が必要。
- 内部データの都合上、リグヒ専用スキル（サンライズヒーローなど）も含まれている。
- hp_per / order / order_rate が固定条件へ再分類されたが、rr フィールドへの格納は維持されている。calcUnc の FIXED_RR_VARS による除外ロジックが正しく動作することを前提とする。
- v4.1 の外部JSファイルは var 宣言を使用するため、厳格モード（'use strict'）と組み合わせる際は注意が必要。