Quantitative Analysis of RuneScape 3 Combat

Akritia[a], Kyroh[a], Gamedolf[c], Sfox[c]

January 4, 2024

RS Analysis, RS Math Discord, and The PvM Encyclopedia

Abstract

RuneScape Player versus Monster (PvM) encounters are fundamentally static. In most cases, Non-Player Character (NPC) behavior is structured the same across all instances of an encounter. Players execute predetermined sequences of abilities called rotations which have been optimized for speed and consistency. The PvM Encyclopedia offers publicly available rotations for every boss, although they are primarily human-generated through trial and error. We propose that RuneScape can be solved and we explore the potential of statistical methods to evaluate both individual actions as well as complete rotations. Player-derived damage in a rotation can be interpreted as discrete random variables with non-identical distributions. We find that the distribution of sequences of abilities obtain Gaussian characteristics over time and show that sufficiently long rotations can be approximated with a Gaussian Probability Mass Function (PMF). These methods are useful for comparative analysis of existing rotations. However, we aim to transcend intuition-based rotation optimization through reinforcement learning–and briefly examine the mathematical landscape of solving stochastic Markov Decision Processes (MDPs) for massively large spaces in the context of RuneScape combat.

> *"No one in the universe looks at RuneScape and says, you know what the most appealing part of this game is? The damage formula that requires 3 Ph.D.'s and a government research grant to understand."* –Stelaro

[a] authors, [c] contributors

Contents

List of Figures

List of Tables

1 Introduction

RuneScape is a massively-multiplayer online role playing game (MMORPG) published by Jagex in 2001. Over the years, the game has changed substantially with frequent updates to the skills, quests, and much more. Perhaps the most substantial update in the history of RuneScape was the Evolution of Combat or "EOC" on November 20th, 2012[5]. EOC established a fundamentally new framework for combat mechanics by introducing *abilities*, *adrenaline*, and *cool-downs*. On the surface, the new combat system was relatively simple; the different combat styles had books of abilities that could be *cast* with some outcome. Casting an ability either generates or costs adrenaline and initiates a *global cool-down (GCD)* where the player cannot cast most other abilities for a universal time frame. Under the surface, this new combat system introduced a labyrinth of complex interactions that players have tried to map out for the past 11 years.

RuneScape is hypothetically solvable, meaning perfect play is potentially achievable unlike other real-time combat systems. This is primarily because of two constraints within the game's mechanical framework: The first is the "tick," a 0.6 second time cycle that is the minimum interval where the game state can change. Most games use a tick size that are a fraction of this, but the comparatively larger tick size allows for a highly generous margin of error for user inputs; second, the boss encounters are generally static in their behavior–the mechanical pattern of a boss is the same between any two kills. Because of these constraints, players can plan and execute a sequence of abilities–we refer to these sequences as rotations–designed to complete a boss encounter in a game theory optimized way. We aim to provide analytical methods that can be used for comparing currently existing rotations and to give an overview of how dynamical programming can be used to discover new optimal rotations.

2 Basic Combat Mechanics

Before showcasing the mathematics of rotation optimization we must first explore the complex combat landscape of RuneScape to gain an understanding of the mechanical framework of abilities, critical hits, and damage calculations. RuneScape's combat system revolves around four combat styles: magic, ranged, melee, and necromancy. The combat triangle encompasses magic, ranged, and melee, sharing several commonalities, while necromancy exists outside the combat triangle and is mechanically different from the combat triangle styles. The player's equipped main-hand weapon dictates their chosen combat style, granting access to style-specific ability books.

2.1 The Combat Triangle

Within the combat triangle, abilities are categorized into three main types: basics, thresholds, and ultimates. Basics serve as adrenaline-generating abilities, dealing minor damage and occasionally featuring auxiliary effects such as damage boosts, critical strike chance buffs, or stuns. Thresholds cost 15% adrenaline and require 50% adrenaline to cast; they usually deal more damage than basics and can offer similar auxiliary effects. Ultimates come at a high adrenaline cost, 100% as a baseline, but can be subject to cost reductions through various means. They often deal substantial damage or provide significant duration-based auxiliary effects.

Abilities are the primary mechanism for dealing damage; in most cases, these abilities are cast in intervals of three ticks, one GCD, although there are some instances where that is not the case. For example, channeled abilities, abilities where each hit has a charge time and if interrupted will cancel the remaining hits–frequently have cast times longer than a GCD, frequently have cast times longer than a GCD but can be canceled prematurely by inputting another ability.

Abilities are not the only damage mechanism that exist within rotations. Two other player-derived damage mechanisms include weapon specials (spec) and auto attacks (auto). In most cases, weapon specs are mechanically similar to abilities wherein they are typically cast on GCD, deal damage, and cost an amount of adrenaline specific to the weapon specs. Auto attacks are different; autos are cast at a frequency determined by the equipped weapon when no abilities are cast and do not initiate a GCD.

2.2 Necromancy

Necromancy was released on August 7th, 2023, and is a fundamentally new mechanical framework for RuneScape combat. There are a few distinctions to introduce before discussing the mechanical framework of the combat style. Necromancy is the first style to scale to level 120, which is a considerable increase in base damage because level is one of the core elements of ability damage. In addition, the skill gets considerably more value out of percentage-based level boosts because of the higher level ceiling. Along the way to level 120, players unlock progressive critical hit damage modifiers that act as the basis of the new critical strike framework covered in detail in section 6.

There are three main player actions for necromancy: abilities, conjures, and incantations. Unlike the combat triangle styles, every player action initiates a GCD, even auto-attacks. Damage for necromancy is based on two largely independent operating mechanisms: necrotic attacks and conjures. necrotic attacks are abilities categorized as either basics, ultimates, or other that deal necromancy damage within a defined range. The other abilities have distinct effects, damages, and adrenaline costs. Basics are also quite different for necromancy because there is not a large selection of basic abilities to create a unique basic rotation. Necromancy's auto-attack functions as its main adrenaline generating ability, all basic abilities do the same damage but with added auxiliary effects. These effects create an interesting decision landscape for the player to find the highest-value option in a given scenario.

The other damage mechanism within the necromancy style are conjures; they are minions summoned by the player and fight alongside the player, dealing spirit damage, which is not directly sourced by the player. The conjures continuously send their auto attacks to the enemy target at a discrete frequency for the duration of the conjure. Each conjure has a secondary activation that can have various effects.

Beyond direct damage mechanics, necromancy has a spellbook of incantations. Each has a unique effect, granting some utility to the player for various combat scenarios.

3 The Ability Damage Conjecture

In RuneScape, the damage dealt by an attack is non-deterministic, meaning the hits are randomly determined within a minimum and maximum bound. These ranges are denoted on tooltips as a percentage derived from the base stat known as ability damage (A_d). For instance, Omnipower, as depicted in Figure 1, inflicts damage ranging between 200% and

400% of A_d. RuneScape does not disclose the equations for calculating ability damage, leaving players to speculate its formulation.

Figure 1: Omnipower tooltip

3.1 Dissecting ability damage

Through experimentation–likely involving swapping gear, weapons, or consuming potions–players have discerned that A_d consists of three additive elements: level (l), damage tier (t), and armour bonus (b).

Level (l) represents the boosted level derived from all currently active stat-boosting effects in the combat skill for the associated worn weapon. Damage tier (t) is determined by selecting the lower value between the equipped tier(s) of the player's weapon(s) and any associated ammunition's (ammo) tier. In the context of magic, the term "ammo" refers to spell tier, while for ranged combat, it pertains to bolt or arrow tier. The calculation considers only the weapon tier for melee and necromancy, which do not require ammunition. The final component, armour bonus (b), is an aggregate value derived from all style bonus attributes associated with the player's currently equipped gear. These style bonus values are displayed on equipment tooltips as "Damage Bonus." The reaper crew passive bonus is applied as an armour bonus and, for now, is the only non-equipment-based style bonus in the game.

3.2 Ability damage equations

Each additive element, l, t, b, has a coefficient to modify the element's weight on A_d. For many years, players believed the A_d equations to be one of three functions depending on worn weapon type where

$$A_d = \begin{cases} \lfloor 2.5l \rfloor + \lfloor 9.6t \rfloor + \lfloor b \rfloor & \text{main-hand} \\ \lfloor 1.25l \rfloor + \lfloor 4.8t \rfloor + \lfloor 0.5b \rfloor & \text{off-hand} \\ \lfloor 3.75l \rfloor + \lfloor 14.4t \rfloor + \lfloor 1.5b \rfloor & \text{two-hand} \end{cases} \tag{1}$$

Equation(1) correctly captures several fundamental truths of A_d. Namely, the damage tier carries the most weight, followed by level and armour bonus. Furthermore, the sum of the coefficients for main-hand and off-hand weapons equals the coefficients of two-hand weapons, with two-thirds weighted on the main-hand. Applying these equations broadly, some inexplicable discrepancies appear. As an example, Table 1 is one edge case we discovered wherein Equation (1) produces results inconsistent with the in-game A_d.

We formulated various potential A_d equations and, for every iteration, performed tests to uncover potential discrepancies between the output of the iterative equation and the

Table 1: Discrepancies in A_d between Equation (1) and data from the game.

Weapon(s)	Eq.(1)	Game
Tier 92 two-hand	1712	1712
Tier 92 dual-wield	1712	1713

in-game value. More often than not, for a given iteration of the equations, there is a particular set of parameters capable of producing a discrepancy. We eventually found equations that appear to be consistent across all observable instances.

$$A_d^{\text{Magic}} = \begin{cases} \lfloor 2.5l \rfloor + \lfloor 9.6t + b \rfloor & \text{main-hand} \\ \lfloor 0.5 \cdot (\lfloor 2.5l \rfloor + \lfloor 9.6t + b \rfloor) \rfloor & \text{off-hand} \\ \lfloor 2.5l \rfloor + \lfloor 1.25l \rfloor + \lfloor 14.4t + 1.5b \rfloor & \text{two-hand} \end{cases} \tag{2}$$

$$A_d^{\text{Melee}} = \begin{cases} \lfloor 2.5l \rfloor + \lfloor 9.6t + b \rfloor & \text{main-hand} \\ \lfloor 0.5 \cdot (\lfloor 2.5l \rfloor + \lfloor 9.6t + b \rfloor) \rfloor & \text{off-hand} \\ \lfloor 2.5l \rfloor + \lfloor 1.25l \rfloor + \lfloor 9.6t + b \rfloor + \lfloor 4.8t + 0.5b \rfloor & \text{two-hand} \end{cases} \tag{3}$$

$$A_d^{\text{Ranged}} = \begin{cases} \lfloor 2.5l \rfloor + \lfloor 9.6t + b \rfloor & \text{main-hand} \\ \lfloor 0.5 \cdot (\lfloor 2.5l \rfloor + \lfloor 9.6t + b \rfloor) \rfloor & \text{off-hand} \\ \lfloor 2.5l \rfloor + \lfloor 1.25l \rfloor + \lfloor 9.6t + b \rfloor + \lfloor 4.8t_1 + 0.5b \rfloor & \text{two-hand} \end{cases} \tag{4}$$

$$A_d^{\text{Necromancy}} = \begin{cases} \lfloor 2.5l \rfloor + \lfloor 9.6t + b \rfloor & \text{main-hand} \\ \lfloor 0.5\, (\lfloor 2.5l \rfloor + \lfloor 9.6t + b \rfloor) \rfloor & \text{off-hand} \end{cases} \tag{5}$$

Note that t_1 is the same as t, except when the ammo tier is 0. Then, t_1 is based solely on the tier of the weapon.

4 Iterations of Damage Calculation

RuneScape calculates damage through an iterative process where, at each step, the resulting value is rounded down to a natural number. In most cases, a given iteration of the damage calculation is the product of a damage effect and a base damage value wherein the base damage value can be ability damage A_d, fixed and variable damage A_f/A_v, or the damage roll d. If the damage effect applies to $A - d$ directly, it is called on-cast; if it applies to fixed and variable damage, it is called on-hit; and if it applies to the damage roll that will be sent to the NPC, it is called on-NPC.

$$\xrightarrow{A_d} \boxed{\text{On-cast}} \xrightarrow{A'_d} \boxed{\text{On-hit}} \xrightarrow{\begin{bmatrix} A_f \\ A_v \end{bmatrix}} \boxed{\text{On-NPC}} \xrightarrow{d}$$

Figure 2: Diagram of general damage stages

4.1 On-Cast

If x_1 is an on-cast multiplicative percentage damage boost, then the new iteration of A_d can be calculated as:

$$A'_d = A_d + \lfloor A_d \cdot x_1 \rfloor .$$

When there are multiple effects to be applied, x_1 and x_2, the calculation becomes

$$A'_d = A_d + \lfloor A_d \cdot x_1 \rfloor + \lfloor (A_d + \lfloor A_d \cdot x_1 \rfloor) \cdot x_2 \rfloor,$$

the resulting change in ability damage for each effect is rounded down and added to ability damage. Therefore, on-cast formulation can be thought of as an iterative process of A_d calculation that can be abstracted to n effects

$$A_n = A_{n-1} + \lfloor A_{n-1} \cdot x_n \rfloor \tag{6}$$

where A_n represents the n^{th} iteration of A_d.

4.2 On-Hit

If an effect is on-hit, it applies to the fixed and variable damage portions independently. Fixed damage, A_f, is the minimum damage an ability can deal, calculated using the minimum damage percentage of the ability a_f and the final ability damage A'_d where

$$A_f = \lfloor a_f \cdot A'_d \rfloor. \tag{7}$$

Variable damage, A_v, is the random damage an ability can deal and calculated as the difference between minimum and maximum damage percentage, a_f and a_m, where

$$A_v = \lfloor (a_m - a_f) \cdot A'_d \rfloor. \tag{8}$$

Let us take Omnipower from Figure 1 as an example when $A'_d = 2000$,

$$A_f = \lfloor 2.0 \cdot 2000 \rfloor = 4000,$$

$$A_v = \lfloor (4.0 - 2.0) \cdot 2000 \rfloor = 4000.$$

The proportionality of fixed and variable damage is important; in this case, $f \propto v = 1$. However, we find that in cases where $f \propto v = \frac{1}{4}$, the fixed and variable damage is calculated from a pseudo max hit m_p instead of A'_d where

$$\begin{aligned} m_p &= \lfloor a_m \cdot A'_d \rfloor, \\ A_f &= \lfloor a_f \cdot m_p \rfloor, \\ A_v &= \lfloor (a_m - a_f) \cdot m_p \rfloor. \end{aligned} \tag{9}$$

General multiplicative percentage boosts are calculated in a peculiar way, the game uses a base value of $\varepsilon = 10000$ then applies each effect in the process to that value. The n^{th} iteration of on-hit effects to ε is

$$\varepsilon_n = \varepsilon_{n-1} + \lfloor \varepsilon_{n-1} \cdot x_n \rfloor, \tag{10}$$

similar to on-cast effects. If ε' is the final iteration of ε, meaning all effects have been applied, then we obtain the final fixed and variable damage as

$$A'_f = \frac{A_f \cdot \varepsilon'}{10000}, \tag{11}$$

$$A'_v = \frac{A_v \cdot \varepsilon'}{10000}. \tag{12}$$

Most effects apply as expected, however, some formulations of on-hit are different.

1. **Additive effects:** Certain effects that apply during on-hit are additive instead of multiplicative wherein the two additive effects, x_1 and x_2, would be calculated as $d_n = d_{n-1} + \lfloor d_{n-1} \cdot (x_1 + x_2) \rfloor$.
2. **Damage over time (DoT) exclusion:** DoT abilities are not impacted by on-hit effects; they skip this calculation subsection entirely.
3. **Fixed and variable damage differences:** Some on-hit effects do not apply equally to both fixed and variable. Some effects apply inversely, disproportionately, or skip fixed or variable damage entirely.

4.2.1 Damage per Level (DPL)

Players discovered that abilities, at some point in the on-hit calculation, receive a flat amount of damage per boosted level (DPL). Here, Δb represents the number of currently boosted levels. The equations are as follows:

$$A'_f = \lfloor A_f + (4 \cdot \Delta b) \rfloor,$$

$$A'_v = \lfloor A_v + (4 \cdot \Delta b) \rfloor.$$

DPL does not apply to Necromancy damages at all. Additionally, there are some instances where the net $8\times$ multiplier for boosted levels applies to either fixed or variable damage, rather than split evenly between the two; those that are known are clarified in section 7.

4.2.2 Invention perks

The precise and equilibrium perks have an inverse relationship between fixed and variable damage. Precise increases the minimum hit of an attack by 1.5% per rank r of the attack's maximum hit A_m. Because the minimum hit is increased, without a maximum hit increase, variable damage must be reduced by an equal amount; therefore

$$A'_f = A_f + \lfloor r \cdot 0.015 \cdot A_m \rfloor,$$

$$A'_v = A_v - \lfloor r \cdot 0.015 \cdot A_m \rfloor.$$

Similarly, equilibrium increases the minimum damage of an attack by three percent and decreases maximum damage by one percent of variable damage,

$$A'_f = \lfloor A_f + r \cdot 0.03 \cdot A_v \rfloor,$$

$$A'_v = \lfloor A_v - r \cdot 0.04 \cdot A_v \rfloor.$$

For now, these are the known idiosyncrasies of on-hit formulation.

4.3 On-Npc

For effects that are on-npc, the application of effects is to one damage value d, which represents the sum of A_f and a random damage amount $\sim U[0, A_v]$, where $U[a, b]$ is a uniform distribution with equal probability of sampling values between a and b. In the same manner as on-cast, the n^{th} iteration of on-NPC multiplicative boosts can be can be calculated as:

$$d_n = d_{n-1} + \lfloor d_{n-1} \cdot x_n \rfloor. \tag{13}$$

Table 2: An incomplete order of application of effects

On-cast	**On-hit part 1**	**On-hit part 2**	**on-npc**
Chaos roar	Arrow effects	Dominion tower gloves	Kerapac's wristwraps
↓	↓	↓	↓
Hex hunter	DPL	Melee bane weapons	Vulnerability
↓	↓	↓	↓
Greater sonic wave	Pernix quiver	Slayer helm	Smoke Cloud
	↓	↓	↓
	Additive effects	Fort Forinthry guardhouse	Gloves of Passage (bleed)
	↓	↓	↓
	Prayer	Genocidal	X Slayer Perk
	↓	↓	↓
	Damage boosting ultimates	Salve amulet	X Slayer Sigil
	↓	↓	↓
	Exsanguinate	Ripper claw	Aura boost & Metamorphosis
	↓	↓	↓
	Revenge	Ripper passive	Scrimshaws
	↓	↓	
	Spendthrift	Precise	
	↓	↓	
	Ruthless	Equilibrium	

5 Forced Auto-attacks

Auto-attacks from the player within the combat triangle originate from the legacy combat system and occur at a frequency derived from the equipped weapon attributes if uninterrupted by ability inputs. Auto-attacks have a $A_f = 1$ and $A_v = W_d$, where W_d is the weapon damage value of the equipped weapon. They are subject to the same damage modifiers as abilities are.

For any combat triangle style, auto-attacks can be forced by clicking the target while casting an ability immediately following any non-damaging ability. Players have also discovered a more lucrative way to force these auto attacks using the different attack frequencies of weapons. Because magic autos can be cast by player input, there is a method to weave in auto attacks at the cost of one game tick. By switching between dual-wield and two-hand weapons every other ability cast, a two-hand auto which is more powerful than a dual-wield auto, can be cast on the fourth tick of every other GCD. This method is called four tick auto attacking (4TAA) where the foundational attack structure is *Auto + ability* → 3 *ticks* → *dual-wield ability* → 4 *ticks* → *Auto + ability*.

6 Critical Strikes

RuneScape's combat system, like many other RPGs, has a probabilistic critical hit system. In the current state, critical strikes can manifest through forced or natural means. Within the combat triangle, critical hits have a hit cap than non-critical hits. Forced critical hits are rolled before the natural damage roll, and the forced critical hit probability is determined by the sum of all critical hit chance boosting effects. When a forced critical strike occurs, the damage range of the ability is set to the following interval

$$d_{\text{forced}}^{\text{combat triangle}} = A'_f + U[\lfloor 0.95 \cdot A'_v \rfloor, A'_v]. \tag{14}$$

The other critical hit mechanism is natural critical hits that occur when the player fails to get a forced critical hit, but the damage roll (d_r) is such that

$$d_r \geq \lfloor 0.95 \cdot (A_m) \rfloor. \tag{15}$$

The hit is then subject to the same critical strike effects as forced critical strikes.

Necromancy critical strikes are mechanically different from the combat triangle styles. There is a base forced critical hit chance $P(f) = 0.10$ with no mechanism for natural critical hits. The damage roll and forced critical hit roll occur independently; when a forced critical hit is rolled, the damage of the critical strike d_f then is given by

$$d_{\text{forced}}^{\text{Necromancy}} = \lfloor d_r \cdot C_d \rfloor, \tag{16}$$

where d_r is the value determined by the damage roll, and C_d is the associated critical hit damage modifier. This system significantly impacts the shape of the damage distribution of necromancy abilities, further discussed in section 8.

7 Ability Oddities

There are widespread idiosyncrasies in RuneScape's combat landscape, we will briefly discuss all of those of which we are aware.

7.1 Corruption Blast/Shot Damage Over Time

The tool tip states that it deals 33% to 100% A_d, reducing by 20% per hit following the initial hit. We find the mechanics to be that the last hit is rolled (6.6% to 20% A_d) and is then multiplied by a scalar to determine the n^{th} hit.

7.2 Tendrils: Smoke, Shadow, Blood

The A_d percentage is a function of the damage dealt to the player and then uses a scalar to determine the hit. Shadow and Smoke tendrils get the full DPL bonus on fixed damage, which is believed to have something to do with the fact that they always land as a critical hit when damage-boosting prayer is active or levels are boosted.

7.3 Crystal Rain (Seren Godbow Special Attack)

Assuming one hundred percent hit chance, the first arrow of the Seren Godbow (SGB) special attack, Crystal Rain, will always hit the target and the damages are calculated as any other hit. The intricacies lie within the auxiliary arrows grid style area of affect (AOE) and the decaying variable damage. First we must show how NPC centering works, an NPC can be thought of as an $n \times n$ matrix where the center for odd n is the true center (C) and for even n is the southwest tile of the 2×2 grid in the center of the matrix. The $\square$ are the rest of the tiles occupied by the NPC. If A_n is an $n \times n$ matrix the centers for $n = 5$ and $n = 4$ are as follows:

$$
A_5 = \begin{matrix} \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square \\ \square & \square & C & \square & \square \\ \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square \end{matrix} \qquad A_4 = \begin{matrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & C & \square & \square \\ \square & \square & \square & \square \end{matrix}
$$

The center of an NPC is chosen and the radius is calculated as the floor of half of the NPC size. A square within the radius from the center of the NPC is chosen as the center of the special attack. A 5×5 area centered at that square is where the remaining four arrows will be chosen. A random square is chosen from this area, and then random squares chosen, without duplicates, until there are four arrows, or when it fails to find a non-duplicate ten times. The number of squares chosen that also lie on the NPC squares are the number of arrows that hit. Using the same examples as above, s marks the tiles that can be randomly determined as the center of the crystal rain matrix and $\not s$ are those which are non-NPC tiles.

$$
A_5 = \begin{matrix} s & s & s & s & s \\ s & s & s & s & s \\ s & s & C_s & s & s \\ s & s & s & s & s \\ s & s & s & s & s \end{matrix} \qquad A_4 = \begin{matrix} \not s & s & s & s & s \\ \not s & s & s & s & s \\ \not s & s & C_s & s & s \\ \not s & s & s & s & s \\ \not s & \not s & \not s & \not s & \not s \end{matrix}
$$

Now let's say that $\odot$ is the center of the crystal rain matrix, $\ominus$ are non-NPC crystal rain tiles, $\oplus$ are NPC and crystal rain tiles, and $\square$ are unaffected NPC tiles.

$$
A_5 = \begin{matrix} & \square & \square & \square & \square & \square \\ \ominus & \oplus & \oplus & \oplus & \oplus & \square \\ \ominus & \oplus & \oplus & \oplus & \oplus & \square \\ \ominus & \oplus & \odot & \oplus & \oplus & \square \\ \ominus & \oplus & \oplus & \oplus & \oplus & \square \\ \ominus & \ominus & \ominus & \ominus & \ominus & \end{matrix} \qquad A_4 = \begin{matrix} & & \square & \square & \square & \square \\ \ominus & \ominus & \oplus & \oplus & \oplus & \square \\ \ominus & \ominus & \oplus & \oplus & \oplus & \square \\ \ominus & \ominus & \odot & \oplus & \oplus & \square \\ \ominus & \ominus & \ominus & \ominus & \ominus & \\ \ominus & \ominus & \ominus & \ominus & \ominus & \end{matrix}
$$

When an arrow lands on a tile in the crystal rain matrix, higher-order arrows can no longer land on that tile. Regarding the damage calculation of the arrows themselves, the variable damage of each additional arrow after the first decays such that the k^{th} arrow is calculated as

$$
\begin{aligned} A_{fk} &= A_{f_1}, \\ A_{vk} &= \left\lfloor \frac{\max(0, 8-k)}{3} \cdot A_{f_1} \right\rfloor . \end{aligned}
$$

Auxiliary arrows from multi-source SGB casts that are 8 arrows or higher have zero variable damage[10].

7.4 Bash

The tooltip states that it deals additional damage equal to 10 percent of your shield's armour value plus defence level. It in fact does not take 10 percent of anything and instead calculates damage as

$$A_f = \lfloor 0.2 \cdot (A_d + l_d + s) \rfloor,$$
$$A_v = \lfloor 0.8 \cdot (A_d + l_d + s) \rfloor,$$

where l_d is defence level and s is shield armour value.

7.5 Deadshot and Massacre

The damage over time portion of these abilities is equal to $\lfloor \frac{m_p}{3} \rfloor$, where m_p is the pseudo max hit of the initial hit. It is still affected by on-npc effects.

7.6 Greater Ricochet

The primary hit of Greater Ricochet is calculated as any other ability with $f \propto v = \frac{1}{4}$. The secondary ($f_2$, v_2) and tertiary (f_3, v_3) hits have

$$f_2 = \left\lfloor \frac{f_1}{2} \right\rfloor,$$
$$f_3 = \left\lfloor \frac{f_2}{2} \right\rfloor,$$
$$v_2 = \left\lfloor \frac{v_1}{2} \right\rfloor,$$
$$v_3 = \left\lfloor \frac{v_2}{2} \right\rfloor,$$

where f_1 and v_1 are the fixed and variable values of the first hit after all on-hit boosts have been applied. All hits roll separately and have on-npc effects applied separately.

7.7 Snap Shot

The variable damage of the second hit of Snap Shot is calculated as scaled damage of the hit one variable damage[11].

$$f_2 = f_1,$$
$$v_2 = \lfloor 1.1 \cdot v_1 \rfloor.$$

7.8 Hurricane

For the second hit of Hurricane, DPL is calculated as

$$f_2' = f_2 + \left\lfloor 10 \cdot \frac{v_2}{v_1} \cdot \Delta b \right\rfloor,$$
$$v_2' = v_2 + \lfloor 2 \cdot \Delta b \rfloor,$$

then precise is calculated as

$$f_2'' = f_2' + \left\lfloor \left(1 + \frac{v_1'}{f_1'}\right) \cdot f_2' \cdot 0.015 \cdot r \right\rfloor,$$
$$v_2'' = \left\lfloor \frac{v_2'}{v_1'} \cdot v_1'' \right\rfloor.$$

The calculation then proceeds as normal[9].

7.9 Perfect Equilibrium (Bow of the Last Guardian Passive)

The minimum damage for the hit derived perfect equilibrium (PE) proc is 25 percent rather than 35 as stated on the tool-tip. For the perfect equilibrium proc itself, different effects are calculated differently depending on the stage that they apply. If the calculation happens pre-DPL it applies to the ability damage derived damage amounts and the hit derived damage amounts independently whereas post-DPL effects it applies to the combined fixed and combined variables damage portions[14].

7.10 Bleeds and Burns

Combust, Fragmentation Shot, Dismember, and Slaughter all roll a value between 1 and m_p and take the max between that roll and the minimum damage percent of the ability. The result is that these bleeds will min roll on average $\frac{min\%}{max\%}$ of casts.

7.11 Shatter

Fixed damage gets the full DPL bonus, similar to tendrils, we believe the guaranteed critical hit is a side effect of this.

8 Statistical Understanding of Damage Values

In this section, we delve into a statistical analysis of damage values in RuneScape, focusing on understanding the nuances and complexities of damage calculation. Our analysis aims to demystify these calculations, clarifying how different elements interact and contribute to the final damage figures observed in gameplay. We explore several key aspects, including the expected value of damage, variance and standard deviation, and the distribution patterns of single-hit and multi-hit abilities. By dissecting these components, we aim to comprehensively understand damage mechanics, the underlying mathematics, and the practical implications.

8.1 Damage as Expected Value

The average damage of an ability μ_a is calculated as the weighted mean of the damage range as opposed to the average between min and max hit because of the weight of forced critical hits. The weighted mean μ_a of an ability a is the expected value given by:

$$\mu_a = E[a] = \sum_{d_a \in a} d_a \cdot P(d_a), \tag{17}$$

where d_a represents a possible damage value in the domain of a and $P(d_a)$ is the probability of the damage value occurring. When discussing RuneScape hits, μ_a, $E[a]$ weighted mean, expected value, or expected damage all describe the exact same value derived from equation (19). The expected damage of multi-hit abilities is represented as the sum of the expected damage for the individual hits μ_h:

$$\mu_a = \sum_i \mu_{h_i}$$

Expected damage should not be interpreted literally; players cannot cast an ability and expect it to deal μ_a damage because it does not capture the variability of potential outcomes. Two abilities with equal μ_a could have very different distributions and are most appropriately evaluated by calculating variance and standard deviation.

8.2 Variance and standard deviation

Variance σ^2 indicates how much the damage values of ability are expected to deviate from the average damage μ_a. For an ability a, the variance is calculated as:

$$\sigma^2 = \sum_{d_a \in a} (d_a - \mu_a)^2 \cdot P(d_a)$$

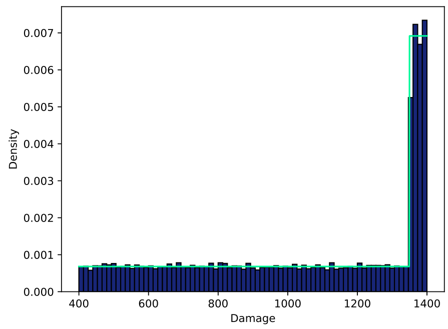
where d_a is a possible damage value, μ_a is the weighted mean of the damage, and $P(d_a)$ is the probability of that damage occuring. The standard deviation σ is the square root of the variance $\sqrt{\sigma^2}$ and provides a more intuitive measure of variability because it is in the same units as the damage itself.

8.3 Distributions of Single-Hit Abilities

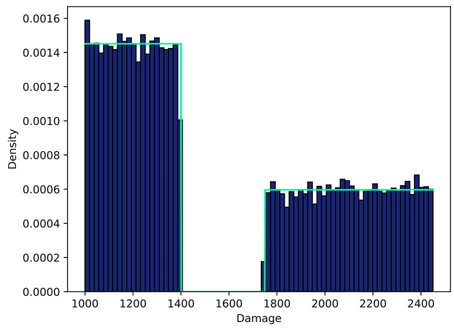
The damage roll of a single hit ability is discrete and uniform; any damage value in a particular domain is equally likely to occur. The most appropriate way to present this mathematically is to say that the damage of an ability can exist within two domains– λ_{natural} for a natural roll and λ_{forced} for a forced critical strike. There may also be instances of a third domain where the damage value can occur as either a natural roll or a forced critical hit in which the probability of its likelihood is the sum of the two probabilities. The probability mass function (PMF) of an ability $p_a(d_a)$ then would be

$$p_a(d_a) = \begin{cases} P(\text{natural}) \cdot \lambda_{\text{natural}}^{-1} & d_a \in \lambda_{\text{natural}} \text{ and } d_a \notin \lambda_{\text{forced}} \\ P(\text{natural}) \cdot \lambda_{\text{natural}}^{-1} + P(\text{forced}) \cdot \lambda_{\text{forced}}^{-1} & d_a \in \{\lambda_{\text{natural}}, \lambda_{\text{forced}}\} \\ P(\text{forced}) \cdot \lambda_{\text{forced}}^{-1} & d_a \notin \lambda_{\text{natural}} \text{ and } d_a \in \lambda_{\text{forced}} \\ 0 & d_a \notin \{\lambda_{\text{natural}}, \lambda_{\text{forced}}\} \end{cases} \tag{18}$$

The distribution produced by the PMF in equation (20) for a combat triangle style and necromancy are very different by virtue of the necromancy critical hit mechanics. The distributions of a hypothetical single hit ability for both damage systems are shown in Figure 3.



(a) Combat triangle styles



(b) Necromancy

Figure 3: $p_a(d_a)$ for combat triangle styles and Necromancy critical hit systems.

8.4 Distributions of Non-Deterministic Multiple-Hit Abilities

For channeled abilities, the hits $[h_1, h_2, h_3, \cdots h_n]$ are rolled independently. The weighted average μ, standard deviation σ, and the PMF $p(a)$ of a specific hit are calculated using the same methods as single hit abilities–they are mechanically identical. The weighted average across all hits of the channeled ability is the sum of the weighted averages for each hit, the standard deviation is the square root of the sum of individual variances, and the PMF p_a is the convolution of the individual PMFs[8]. A channeled ability with two hits a_x and a_y has a distribution produced by the convolution of the PMFs $(p_x(x) * p_y(y))$ defined as:

$$p(a) = p_x(x) * p_y(y) = \sum_{k=-\infty}^{\infty} p_x(k) p_y(a-k) \tag{19}$$

Instead of performing convolution on the PMFs for each hit, we can evaluate the probability vectors produced by the PMFs $p_x = [p_1, p_2, p_3, \cdots, p_n]$ and $p_y = [p_1, p_2, p_3, \cdots, p_n]$. Utilizing the Convolution theorem of the Fourier Transform, the Discrete Fourier Transform (DFT) for our vectors is defined as:

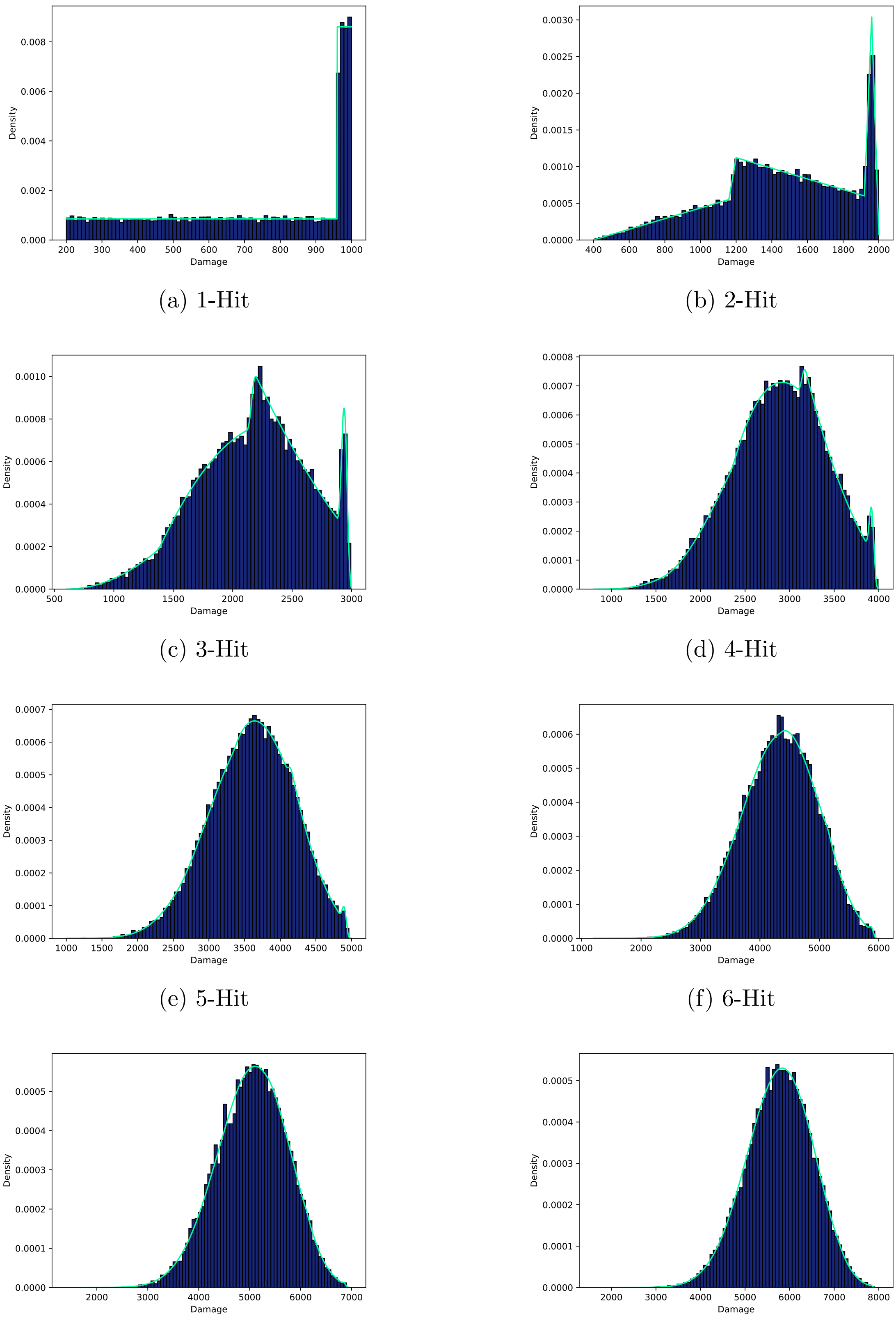
$$\hat{p}_x(\xi) = \sum_{n=0}^{N-1} p_x e^{-i(\frac{2\pi}{N})\xi p_n}$$

$$\hat{p}_y(\xi) = \sum_{n=0}^{N-1} p_y e^{-i(\frac{2\pi}{N})\xi p_n}$$

Then $\hat{p}_a$ is the product of $\hat{p}_x$ and $\hat{p}_y$, we can the find the probability vector for p_a as the Inverse Discrete Fourier Transform(IDFT) of $\hat{p}_a$

$$p_a = \frac{1}{N} \sum_{k=0}^{N-1} \hat{p}_a(\xi) e^{i(\frac{2\pi}{N}\xi p_n)}$$

The resulting vector p_a is the probabilities of the damage vector λ_a for a two-hit ability. The probability vector for a k hit ability is calculated by recursively performing these Fourier transforms for the resulting vector and the next hit.



(a) 1-Hit (b) 2-Hit

(c) 3-Hit (d) 4-Hit

(e) 5-Hit (f) 6-Hit

(g) 7-Hit (h) 8-Hit

Figure 4: Visual transformation of λ_a over 8 convolutions of rapid fire hits.

9 Approximating Damage Distributions

The damage distributions for independent channeled ability hits are mechanically identical to single-hit abilities. Therefore, the PMF of n abilities we add to a rotation r can similarly be evaluated using Fourier transforms. However, convolution runs in log-linear time, which grows considerably as the rotation length increases. A viable alternative when analyzing long rotations is to approximate the damage distribution.

9.1 Gaussian Characteristics in Infinitely Long Rotations

Let us observe what happens to the distribution when the rotation length becomes infinitely long. Suppose r is a rotation of one ability a normalized such that.

$$r = \frac{1}{\sigma_a \sqrt{n}} \sum_{k=1}^{n} (a_k - \mu_a)$$

By virtue of the central limit theorem (CLT), we expect the convergence of r to be normally distributed, $N(0, 1)$. A standard proof of classical CLT utilizes the characteristic function for the ability defined as the Fourier transform of its PMF[6].

$$\phi_a(t) = \sum_{k=1}^{n} e^{ita_k} p_{a_k} = E[e^{ita_k}]$$

Recall that the Fourier transform of p_r is equal to the product of the Fourier transform of each p_{a_k} as discussed in section 8. The characteristic function shares this relationship. If all a_k are identical, then

$$\phi_r\left(\frac{t}{\sqrt{n}}\right) = \left(E\left[e^{i\frac{t}{\sqrt{n}}a}\right]\right)^n$$

Using Taylor's theorem for the polynomial expansion of $e^{i\frac{t}{\sqrt{n}}a}$

$$E\left[e^{i\frac{t}{\sqrt{n}}a}\right] = E\left[1 + i\frac{t}{\sqrt{n}}a - \frac{t^2}{2n}a^2 + \dots\right]$$

The higher-order terms become negligible as $n \to \infty$ therefore

$$\begin{aligned}
\phi_r\left(\frac{t}{\sqrt{n}}\right) = \left(E\left[e^{i\frac{t}{\sqrt{n}}a}\right]\right)^n &\approx \left(E\left[1 + i\frac{t}{\sqrt{n}}a - \frac{t^2}{2n}a^2\right]\right)^n \\
&= \left(1 + i\mu_a\frac{t}{\sqrt{n}} - \frac{t^2}{2n}\sigma_a^2\right)^n \\
&= \left(1 + i(0)\frac{t}{\sqrt{n}} - \frac{t^2}{2n}(1)\right)^n \\
&= \left(1 - \frac{t^2}{2n}\right)^n \\
\lim_{n\to\infty} \phi_r\left(\frac{t}{\sqrt{n}}\right) &= \left(1 - \frac{t^2}{2n}\right)^n = e^{-\frac{t^2}{2}}
\end{aligned}$$

Indicating that a rotation comprised of infinitely many casts of the ability a converges approximately to $N(0, 1)$. Continuously casting a single ability is far from optimal; in most circumstances, we look at non-identically distributed variables. A requisite parameter of classical CLT is that the independent variables are identically distributed[6]. However, we find that non-identically distributed abilities in RuneScape converge similarly to identical distributions[2]. The convergence of $\phi_r(t)$ to the Gaussian form in the non-identical case can be evaluated by checking Lyapunov's condition that requires for some $\delta > 0$

$$\lim_{n\to\infty} \frac{1}{\sigma_r^{2+\delta}} \sum_{j=1}^{n} E\left[\left|a_j - \mu_{a_j}\right|^{2+\delta}\right] = 0 \tag{20}$$

, where $E\left[\left|a_j - \mu_{a_j}\right|^{k}\right]$ gives the k^{th} raw moment of the distribution[6]. For RuneScape rotations, we evaluate Lyapunov's condition at $\delta = 1$ and $\delta = 2$, which calculate skew and kurtosis, respectively. High-order statistics for $\delta > 2$ are not considered because rotations have relatively simple shape parameters and lack the excess degrees of freedom for precise approximations. The 3^{rd} and 4^{th} standardized moments of a Gaussian are zero, and therefore, to confirm the convergence of a rotation to a Gaussian, we expected these order statistics to approach zero.

9.2 Evaluating Convergence with Moments

Given the conditional behavior of p_r in the limit $n \to \infty$, we can instantiate Gaussian approximations for sufficiently small skew γ and kurtosis κ, thereby avoiding the complexity of convolution[4]. We utilize the moment generating function (MGF) of r where

$$\psi_r(t) = E[e^{tr}] = \sum_{r_i} e^{tr_i} p_r(r_i) \tag{21}$$

Skewness γ and kurtosis κ are the third and fourth derivatives of the MGF, respectively, evaluated at $t = 0$.

$$\gamma_r = \dddot{\psi}_r(t)\Big|_{t=0}$$

$$\kappa_r = \ddddot{\psi}_r(t)\Big|_{t=0}$$

The raw moment for kurtosis in a true Gaussian is three, which is standardized by simply subtracting three to obtain excess kurtosis (standardized moment) κ' of zero[4]. For example, let us calculate the convergence of a rotation r when it comprises *Greater Concentrated Blast → Wild Magic → Auto + 3-hit Asphyxiate* where $A_d = 2000$, $W_d = 1000$, and $P(f) = 0.318$. Note that Greater Conc. increases critical strike chance per hit, so the averages of the later hits of the ability and the first hit of Wild Magic have more skew than the other hits. Using the values in Table 2, we can evaluate the convergence of γ_r and

Table 3: Ability data

Ability	Minimum	Maximum	μ	σ
Greater Conc.	1068	5340	3957.90	801.35
Wild Magic	2000	8600	6532.06	1499.60
Auto	0	1000	651.05	325.50
3-hit Asphyx.	2256	11280	8130.79	1695.06

Table 4: Convergence of γ_r and κ'_r

Hit	γ_r	κ'_r
Greater Conc.	-0.38	-0.36
Wild Magic	-0.37	-0.31
Auto	-0.35	-0.29
Asphyxiate	-0.23	-0.17

κ'_r between the true PMF and the Gaussian approximation with each additional hit. As shown by the results in Table 4, γ and κ' are strictly decreasing. The exact moment that these measurements have become sufficiently close to zero for a Gaussian approximation is largely determined by the desired degree of accuracy of the user. Regardless, we can examine this convergence visually in Figure 5.

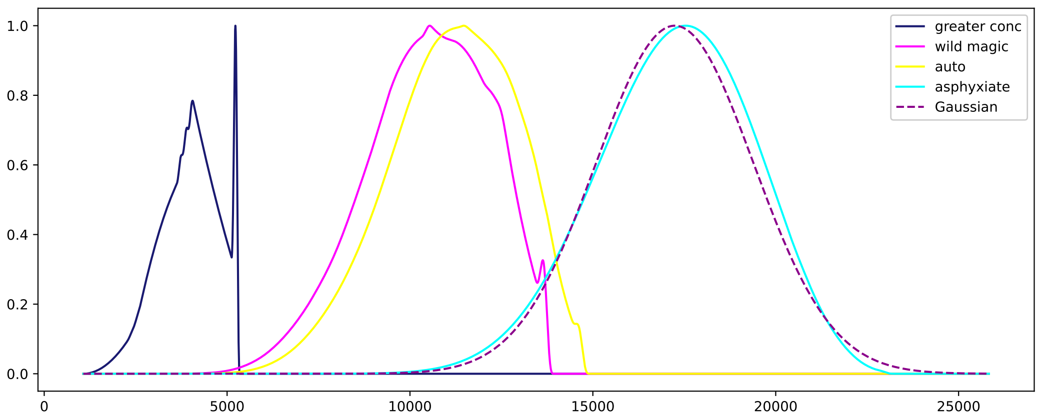


Figure 5: Visual convergence of $p_r(d)$ to a Gaussian $p_g(d)$ with each added ability. Convergence is highly variable; for example, necromancy style abilities with bi-modal distributions take much longer to converge than the combat triangle styles. The convergence should be evaluated for the specific rotation rather than universally applied after n hits.

10 Comparative Rotation Analysis

10.1 Continuous case

We can utilize the Gaussian approximations established in the previous section to evaluate the damage output of a nested rotation. For example, if a player designed a rotation and wants to know how likely they are to hit some damage threshold, we calculate it using a continuous cumulative distribution function[3]. Assume that this rotation r_x has a damage vector λ_x where ψ_x produced $\gamma_x = 0.05$ and $\kappa'_x = 0.04$. Given that the skewness γ_x and excess kurtosis κ'_x for rotation x are close to zero (indicating a near-Gaussian distribution), we can assume a Gaussian distribution with mean μ_x and standard deviation σ_x. The probability density of a particular damage value is given by the PDF p_x

$$p_x(d_x) = \frac{1}{\sigma_x\sqrt{2\pi}} e^{-\left(\frac{d_x-\mu_x}{2\sigma_x}\right)^2} \tag{22}$$

Furthermore, the probability of dealing at least n damage can be evaluated as

$$P_x(d_x \geq n) = 1 - \Phi_x(n)$$

where d_x is a particular damage value in λ_x and Φ_x is the cumulative distribution function (CDF) of r_x defined as

$$\Phi_x(n) = \int_{-\infty}^{n} p_x(t)dt \tag{23}$$

Using these methods, we can also analyze the comparative damage output of two rotations, r_x and r_y. The probability that $d_x > d_y$ is determined by the difference between their respective Gaussian distributions. Let r_x and r_y be two rotations with means μ_x, μ_y and standard deviations σ_x, σ_y, respectively. The random variable $d_z = d_x - d_y$ represents the damage difference between the two rotations. Since r_x and r_y are assumed to be independent and Gaussian, r_z is also Gaussian with mean $\mu_z = \mu_x - \mu_y$ and variance $\sigma_z^2 = \sigma_x^2 + \sigma_y^2$. Thus, the probability that d_x exceeds d_y ($d_z > 0$) can be computed using the cumulative distribution function (CDF) of r_z evaluated at zero

$$P(d_z > 0) = 1 - \Phi_z(0) = 1 - \int_{-\infty}^{0} \frac{1}{\sigma_z\sqrt{2\pi}} e^{-\frac{(t-\mu_z)^2}{2\sigma_z^2}} dt$$

10.2 Discrete case

In many instances, a player may want to evaluate the effectiveness of shorter rotations–only a few abilities in length–which would require the calculation of the discrete PMF rather than the continuous Gaussian approximation. In the discrete case, a rotation r_x has a damage vector $\lambda_x = [x_1, x_2, \dots]$ with probability $p(x_i)$. The CDF then would be

$$\Phi_x(d_x) = P(d_x \geq n) = \sum_{x_i \geq n} p(x_i) \tag{24}$$

Like the continuous case, the probability of one rotation outdamaging another is given by the difference $d_z = d_x - d_y$ evaluated using the discrete CDF Φ_z for $d_z \geq 0$. Comparing the raw damage distributions for rotations is most valuable when they have similar time intervals. For more nuanced analysis, alternative measurements such as damage per tick (dividing damage by the rotation length in ticks) or damage per adrenaline can be considered.

11 The Dynamic Programming Approach

The rotation analysis in section 10 details how we can quantitatively assess the comparative damage output of different rotations. We can improve the rotations with some basic heuristics, however, we do not have any empirical evidence that a rotation is the most optimal option.

11.1 The Bellman Equation for Deterministic Sequences

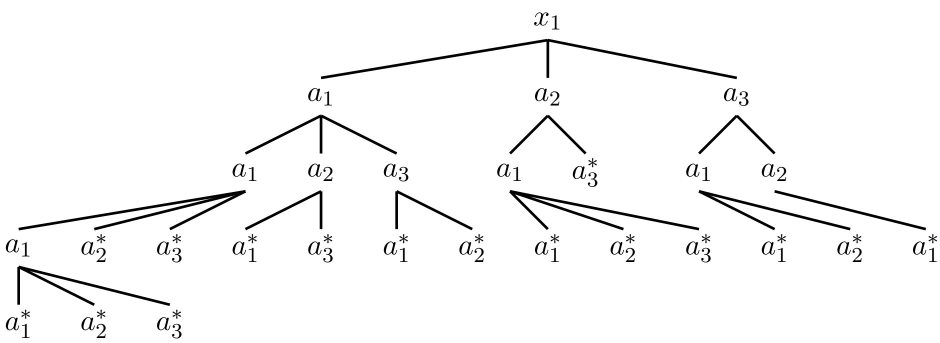
Finding the optimal sequence of actions starts with a brute force approach where a given state x_t is solely determined by the action a_t of the x_{t-1} state. For RuneScape a state would be the status of target's remaining lifepoints, abilities on cooldown, and any other parameters specific to single point in time. Let us start with a simple deterministic landscape where we have three abilities $[a_1, a_2, a_3]$ with the following parameters

Ability	Damage	Cooldown
a_1	10	0
a_2	15	1
a_3	20	2

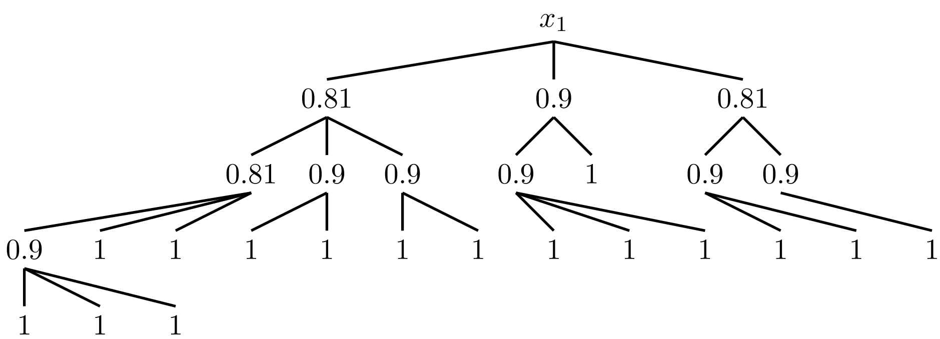
The cooldown denotes the number of states that must pass before an ability can be cast again. The Bellman equation helps us evaluate each potential action's utility relative to each state's advantage or disadvantage[1].

$$V(x_t) = \max_a(U(x_t, a_t) + \beta V(x')) \tag{25}$$

Where $V(x_t)$ is the value function, $U(x_t, a_t)$ is the reward gained by taking an action a_t in a state x_t, and $\beta V(x')$ is the β discounted utility of $V(x_{t+1})$. When $U(x_t, a_t) = 1$, there is an action a_t in a given state x_t capable of bringing the sequence to the terminal state, otherwise $U(x_t, a_t) = 0$. We start with a tree of all state-action pairs and use backward induction to compute the utility gained from each action in the state x_t[1]. If we reach the terminal state when the sum of the damage dealt by all abilities is at least 50, then we obtain:



where $*$ represents a terminal state. For a discount factor $\beta = 0.9$ the utility of the actions across all x_t would be



The optimal ability rotation, in this example, would be $a_2 \rightarrow a_3 \rightarrow a_2$ because those actions yield the highest utility in each state transition. We can verify these actions in the diagram because it is the shortest path to a terminal state. This method, called value iteration, uses the value function to iterate over all states and extract the optimal rotation. RuneScape combat is much more complex than this; many probabilistic elements produce a vector of possible states for each state-action pair. We call these complex random state spaces Stochastic Markov Decision Processes (MDPs).

11.2 Stochastic Markov Decision Processes

The problem of solving an MDP for a RuneScape rotation is the exponentially large number of possible rotations. For massive or infinite state-action spaces, we can use policy-based reinforcement learning methods wherein we start with some arbitrary policy and make iterative improvements, meaning that we are only computing actions fixed by the policy[12]. Policy in the context of RuneScape would be the ability rotation, including every action the player takes tick by tick for the duration of the encounter.

11.2.1 Policy Iteration

In order to carry out policy iteration, we first must establish how to evaluate and improve a policy. The value function of some arbitrary policy π from an action x_t, $V^\pi(x_t)$, is similar to the deterministic case, but rather than the beta discounted utility of the x_t' state, we use the expected value across all possible state sequences defined by the policy[12].

$$V^\pi(x_t) = \sum_{x_t' \in X} P_{\pi(x_t)}\left(x_t' | x_t\right)\left[U(x_t, a, x_t') + \beta V^\pi(x_t')\right]$$

$V^\pi(x_t) = 1$ for terminal states, meaning the probability of the target dying to the next action is 1. If we have a policy $V^\pi(x_t)$ that is not the optimal policy, we can make incremental improvements by changing actions. We will need the state-action value function, also called the Q-function $Q^\pi(x_t, a)$, that yields the expected value of an action a–instead of the policy action–and then following the policy $\pi(x_t)$ from that point forward.

$$Q^\pi(x_t, a) = \sum_{x_t' \in X} P_a\left(x_t' | x_t\right)\left[U(x_t, a, x_t') + \beta V^\pi(x_t')\right]$$

If there exists an action a where the Q-function of a is greater than strictly following the policy $\pi(x_t)$, $Q^\pi(x_t, a) > Q^\pi(x_t, \pi(x_t))$, then the policy is strictly improved by adopting a to the policy $\pi(x_t)$, written as $\pi(x_t) \leftarrow a$[12]. Policy iteration is then an iterative process to extract the optimal policy by calculating $V^\pi(x_t)$ for all x_t using policy evaluation. Then, for each state in the space, $x_t \in X$, we adjust the policy π by iteratively checking each possible action in each possible state and adopting the action to the policy if it improves the value of the Q-function. Written as:

$$\pi(x_t) \leftarrow \text{argmax}_{a \in A(x_t)} Q^\pi(x_t, a)$$

Although policy iteration is more cost-effective than value iteration–because there are finite iterations–each iteration has an exponential cost, which does not scale for sufficiently large spaces. Like the Gaussian approximations established in section 9, an effective alternative is to approximate the optimal policy using policy gradients[13].

11.2.2 Policy Gradients

The fundamental idea of a policy gradient method is to optimize the parameterized policy $\pi_\theta(x_t, a_t)$ to maximize the expected return[13]. The objective function, denoted as $J(\theta)$, represents the expected value of the cumulative rewards obtained by following the policy parameterized by θ:

$$J(\theta) = \mathbb{E}[V^{\pi_\theta}(x_t, a_t)]$$

To improve the policy $\pi_\theta(x_t, a_t)$, we compute the gradient of this objective function with respect to the parameter θ. The goal is to adjust the parameters to increase the expected return over time. The gradient ascent update rule is typically employed:

$$\theta \leftarrow \theta + \alpha \nabla J(\theta)$$

Where α is the learning rate, a hyperparameter that controls the size of the θ parameter updates. $\nabla J(\theta)$ is the gradient of the expected return with respect to θ. The key challenge in policy gradient methods is estimating this gradient accurately. One common approach is the REINFORCE algorithm, which uses a likelihood ratio trick[13]. The gradient of $J(\theta)$ can be expressed as:

$$\nabla J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T} \nabla \log \pi_\theta(x_t, a_t) \cdot \left(\sum_{t'=t}^{T} U_{t'}\right)\right]$$

Where $\nabla \log \pi_\theta(x_t, a_t)$ is the gradient of the log-probability of taking action a_t in state x_t under policy π_θ. $U_{t'}$ represents the cumulative reward obtained from time step t to the end of the episode, which is used as a baseline. This gradient is approximated through multiple episodes by collecting trajectories and computing sample averages. The resulting update rule becomes:

$$\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T_i} \nabla \log \pi_\theta(x_t^{(i)}, a_t^{(i)}) \cdot \left(\sum_{t'=t}^{T_i} U_{t'}^{(i)}\right)\right)$$

Here, N is the number of sampled trajectories, and T_i is the trajectory length in the i^{th} episode. In the context of RuneScape, we can utilize the update rule to approximate the policy gradient and extract the optimal behavior strategy for a player to defeat an encounter in the most time-efficient manner.

11.2.3 Hypothetical Implementation

The implementation of solving an MDP within RuneScape's combat framework presents significant challenges due to the inherent complexity of the problem. While policy iteration is more straightforward conceptually, it can become prohibitively expensive. Constructing a model that attempts to solve the space with policy gradient methods is undoubtedly the most appropriate for the combat landscape in which we operate. Although constructing the model falls outside the scope of this project, there are a number of obstacles that we have ideas about how they may be overcome.

On the construction of the state space and transitions, for a given policy, we suggest that the time interval t when considering possible states x_t and possible actions a_t be ticks as opposed to a GCD. Even though the state space becomes much larger, it is the most appropriate way to accommodate 4TAA, channeled canceling, hit timings, and other off-GCD actions. Possible actions a_t for off-GCD states is often nothing so the considerably smaller time interval does not have as much of an impact as one might think. The state transition probabilities must be modeled in such a way to accommodate the non-linearity of damage and adrenaline changes–even though damage is generally uniform, there are various complexities that introduce non-linearity. The solution then would be to use some form of Gaussian quadrature to approximate the integrals for each state-action transition distribution.

In terms of the policy gradient methods, an actor-critic implementation would likely be most successful because bootstrapping could be used to avoid sampling the full trajectory like traditional Monte-Carlo methods[7]. The critic helps guide the actor towards making better updates to the policy than a pure policy-based method resulting in lower variance and faster convergence. The main obstacle of an actor-critic model is creating a computationally efficient value function approximation. Konda and Tsitsiklis propose a method by which the critic creates projections of the value function from linearly parameterized approximations of the value function using orthogonal polynomials which could theoretically work in this landscape[7].

Our suggestion in terms of practical implementation is to explore actor-critic reinforcement learning methods and the various approximations that can be implemented to make the problem computationally approachable. Finally, we want to note that even though this is the most accurate way to approach the problem, someone could likely get to 95%+ of optimal with a simple Gaussian approximation of the damage distribution, a set of abilities, and a neural network.

12 Conclusions

Since the release of EOC, RuneScape players have progressively pushed the boundary of player understanding of the mechanical underpinnings of the combat system. Our research delves into the game's complex combat system, disclosing the many of the inherent quirks of the game before exploring the various mathematical methods that can be used to formulate optimal play. We start with the basics from the Evolution of Combat update and explore the nuances of abilities, special attacks, and autos in the combat triangle and necromancy.

A large part of our research evaluates ability damage, revealing how level, damage tier, and armor bonus factor into the damage calculation (A_d). We explore the iterative process through which damage is calculated, including of on-cast, on-hit, and on-npc effects. We also examine the probabilistic nature of damage ranges, highlighting the importance of understanding damage probability distributions for various abilities. We demonstrate how Gaussian approximations and high-order statistics can help track damage distribution changes for increasingly long rotations.

Additionally, we investigate using Markov Decision Processes (MDPs) and policy-based reinforcement learning for the creation of optimal ability rotations. This includes basics of policy iteration and gradients, and the challenges of solving MDPs in massively complex state spaces like RuneScape. In summary, our research provides a thorough analysis of RuneScape's combat system, combining mathematical modeling with practical gameplay. This work is a testament to the depth of RuneScape's combat mechanics and serves as a guide for players and researchers interested in the mathematical and computational elements of game optimization.

12.1 Impending changes

Since we began our research, the combat landscape has changed significantly. The game announced and released the Necromancy skill, a rework of many damage effects and mechanical changes. During the last few months leading up to the publication of our research, RuneScape announced the combat beta. At this time, the beta is in its second update phase, and we anticipate that many of the changes will be pushed to the live game quickly. Therefore, the final thing we would like to do here is provide an overview of what we expect to change soon.

- Necromancy mechanics - The new critical hit mechanics introduced by the release of necromancy are almost certainly going to be adopted by the other styles. One side effect of this system is that every ability and special attack needs damage squishing between the minimum and maximum range so that the minimum critical hit is higher than the maximum natural roll.
- DPL Removal - Necromancy does not get a DPL boost, and along with the Necromancy mechanics change, the combat triangle styles similarly will see the removal of DPL.
- Equilibrium - This will likely change again because the current tuning is slightly low. Regardless, as a result of damage squishing equilibrium became effectively useless. The current proposal is to give a flat 4% damage boost per rank–applied during on-hit–at the cost of removing the ability to land as a critical hit.
- Additive effects - All previously additive effects will be changed to be multiplicative so that damage calculation is more intuitive.
- Fixed and variable proportionality - The fixed and variable damage portions were calculated differently for certain abilities as described in section 4.2; they removed this calculation in the beta so all abilities are handled the same.
- Natural critical hits - The transition to necromancy mechanics largely covers this, but we wanted to reiterate that under the new framework, natural critical hits will no longer exist, and all styles will use the necromancy system discussed in section 6.

13 Acknowledgements

This work was a collaborative effort between community members of RS Analysis, The RuneScape Math Discord, and the PvM Encyclopedia. Many of the foundations of our understanding of the RuneScape combat system comes from the contributions of countless players over the lifetime of the game. We thank the RuneScape Wiki and all of the contributors for providing information about the detail of the combat system and its intricacies. We want to give particular recognition to Mod Sponge, Mod Ryan, Mod Pi, and others at Jagex for answering a seemingly endless amount of questions throughout our research. In addition, the work of Ryyhen and Gamedolf on the creation of YARB is largely one of the reasons why we started this project and we thank them for sharing their code with us to use as a resource.

References

[1] Richard Bellman. “The theory of dynamic programming”. In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515.

[2] David M. Bradley and Ramesh C. Gupta. In: *Annals of the Institute of Statistical Mathematics* 54.3 (2002), pp. 689–700. ISSN: 0020-3157. DOI: 10.1023/a:1022483715767. URL: http://dx.doi.org/10.1023/A:1022483715767.

[3] Julio Deride, Johannes O. Royset, and Fernanda Urrea. “A variational approach to a cumulative distribution function estimation problem under stochastic ambiguity”. 2023. arXiv: 2309.00070 `[math.OC]`.

[4] David Draper and Erdong Guo. “The Practical Scope of the Central Limit Theorem”. 2021. arXiv: 2111.12267 `[stat.OT]`.

[5] “EVOLUTION OF COMBAT: NOW LIVE!”. [Online; accessed 16-August-2023]. 2012. URL: https://secure.runescape.com/m=news/evolution-of-combat-now-live.

[6] Tran Loc Hung. “Classical and non-classical central limit theorems for random sums of independent random variables of a double sequence”. 2023. arXiv: 2307.16570 `[math.PR]`.

[7] Vijay Konda and John Tsitsiklis. “Actor-Critic Algorithms”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[8] Lutz Mattner and Bero Roos. “Maximal probabilities of convolution powers of discrete uniform distributions”. 2007. arXiv: 0706.0843 `[math.PR]`.

[9] Sfox. “Hurricane, Explanation of Hurricane hits - The RuneScape Wiki”. [Online; accessed 26-November-2023]. 2023. URL: https://runescape.wiki/w/Hurricane.

[10] Sfox. “Seren Godbow, Special attack - The RuneScape Wiki”. [Online; accessed 26-November-2023]. 2023. URL: https://runescape.wiki/w/Seren_godbow.

[11] Sfox. “Snapshot, Explanation of arrow damages - The RuneScape Wiki”. [Online; accessed 1-December-2023]. 2023. URL: https://runescape.wiki/w/Snap_Shot.

[12] Nancy L. Stokey, Robert E. Lucas, and Edward C. Prescott. “Recursive Methods in Economic Dynamics”. Harvard University Press, 1989. ISBN: 9780674750968. URL: http://www.jstor.org/stable/j.ctvjnrt76.

[13] Richard S Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

[14] Veggie and Jagex Sponge. “Unpublished discussion”. [Discord conversation in the PvM Encyclopedia]. 2023.