Quantitative Analysis of RuneScape 3 Combat

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

December 26, 2023

RS Analysis, RS Math Discord, and The PvM Encyclopedia

Abstract

RuneScape PvM (Player versus Monster) encounters are fundamentally static. NPC behavior, in most cases, is structured the same across all instances of an encounter. Players execute predetermined sequences of abilities called rotations that are 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 explore the potential of statistical methods to evaluate both individual actions and complete rotations. Player derived damage in a rotation can be interpreted as discrete random variables with non-identical distributions. We find that overtime, the distribution of the sum of these variables obtains Gaussian characteristics and therefore show that sufficiently long rotations can be evaluated with a Gaussian approximation. These methods are useful for comparative analysis of existing rotations, however, we aim to transcend intuition based rotation optimization–with dynamic programming and reinforcement learning methods–to gain a more refined understanding of optimal combat engagement in RuneScape's PvM arena.

"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

Contents

List of Figures

List of Tables

1 Introduction

RuneScape is an MMORPG[1] 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[2]. 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 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.

The fundamental theorem of combat is such that players cast abilities in a sequence that maximizes their damage output. RuneScape can be considered a solved game where perfect play is achievable–unlike other real-time combat systems–largely because of two constraints within the games mechanical framework. The first is the "tick," a unit of time that denotes 0.6 seconds and serves as the minimum interval in which the game state can change. Most games use tick sizes fractions 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 way that is game theory optimal. We aim to provide the mathematical methods that could be used for comparative analysis of existing rotations and others used to discover new optimal rotations.

2 Basic Combat Mechanics

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 sits outside of the combat triangle is mechanically different than the trio styles. Players' equipped weapons dictate their chosen combat style, granting access to style-specific ability books.

2.1 The Combat Triangle

Beginning with 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.

[1]MMORPG is an abbreviation for "Mass Multi-player Online Role Playing Game," which is the broader game genre of RuneScape

[2]Jagex Patch Notes, "Evolution of Combat: Now Live!", November 12th, 2012

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 frequently have cast times that are longer than a GCD but can be canceled prematurely by inputting another ability. We briefly discussed rotations, the collection of abilities listed chronologically by cast time. Rotation is a broad term, the entire sequence of abilities for a long fight may be called a rotation, while the sequence for a single phase of that boss may also be called a rotation.

Abilities are not the only damage mechanism that exist within rotations. There are two other player derived damage mechanisms including weapon specials (specs) and auto attacks. Weapon specs, in most cases, are mechanically similar to abilities wherein they are cast on GCD, deal damage, and cost an adrenaline amount specific to the weapon spec. Auto attacks are quite different, they are attacks that are sent to the target at a frequency determined by the equipped weapon when no abilities are cast and for the combat triangle do not initiate a GCD.

2.2 Necromancy

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 tool tips as a percentage derived from 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.

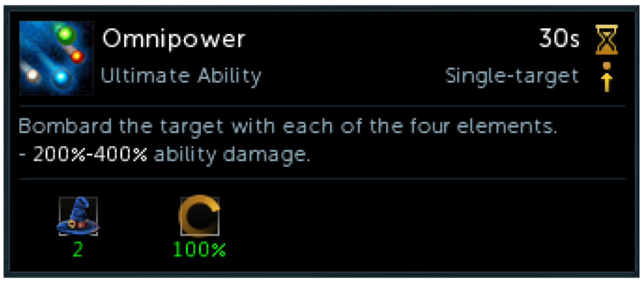
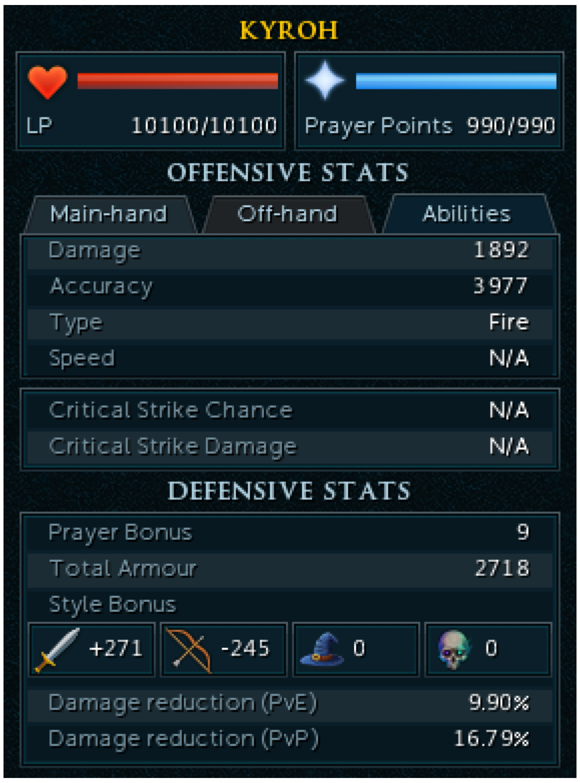

Omnipower
30s
Ultimate Ability
Single-target
Bombard the target with each of the four elements.
- 200%-400% ability damage.

Figure 1: Omnipower tooltip

3.1 Dissecting ability damage

The player's A_d can be viewed through the load out settings tab as shown in Figure 2 which is dynamically updated as A_d changes. 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 tier of the equipped weapon and the tier of any associated ammunition. In the context of magic, the term "ammo" refers to spell tier, while for ranged combat, it pertains to bolt or arrow tier. For melee and necromancy, which do not require ammunition, only the weapon tier is considered in the calculation. 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 tool tips as "Damage Bonus." The application of the reaper crew passive bonus is applied as armour bonus, and for now is the only non-equipment based style bonus in the game.


KYROH

LP 10100/10100

Prayer Points 990/990

OFFENSIVE STATS

Main-hand | Off-hand | Abilities

Damage	1892
Accuracy	3977
Type	Fire
Speed	N/A
Critical Strike Chance	N/A
Critical Strike Damage	N/A

DEFENSIVE STATS

Prayer Bonus	9
Total Armour	2718
Style Bonus	

+271 -245 0 0

Damage reduction (PvE)	9.90%
Damage reduction (PvP)	16.79%

Figure 2: Example $A_d = 1892$

3.2 Ability damage equations

Each of the additive elements, l, t, b, has a coefficient to modify the weight of the element 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, there are some inexplicable discrepancies that appear As an example, Table 1 is one edge case we discovered wherein Equation (1) produces results inconsistent with the in-game A_d.

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

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

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

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

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

$$A_d^{\text{Necromancy}} = \begin{cases} \lfloor 2.5l \rfloor + \lfloor 9.6t + b \rfloor & \text{main-hand} \\ \lfloor 0.5 \left(\lfloor 2.5l \rfloor + \lfloor 9.6t_1 + b \rfloor \right) \rfloor & \text{off-hand} \end{cases} \tag{5}$$

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 uses Ability damage as base damage, it is called on-cast; if it uses fixed and variable damage, it is called on-hit; and if it uses the damage roll that will be sent to the NPC, it is called on-NPC.

$$\begin{tikzcd} {} \arrow[r, "A_d"] & \boxed{\text{On-cast}} \arrow[r, "A_d'"] & \boxed{\text{On-hit}} \arrow[r, "\begin{bmatrix} A_f \\ A_v \end{bmatrix}"] & \boxed{\text{On-NPC}} \arrow[r, "d"] & {} \end{tikzcd}$$

Figure 3: 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 to 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 10,000 to scale up the effect value

$$needtoclarifyhowthisworks \tag{10}$$

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. **Berserk effects:** The additive on-hit effects for melee have their effect halved when berserk is active; these include *Scripture of Ful, Annihilation, Gloves of Passage, and Dragon battle axe.*
3. **Damage over time (DoT) exclusion:** DoT abilities are not impacted by on-hit effects; they skip these calculation sections entirely.
4. **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)

A phenomenon that puzzled players for some time was a discrepancy between external calculations and in-game damage values. For example, say that two single-hit abilities calculated by hand with the same set of parameters result in damage values D_x and D_y; these abilities would differ from the in-game damage values G_x and G_y by a constant k damage.

$$D_x = G_x + k$$

$$D_y = G_y + k$$

$$D_x + D_y = G_x + G_y + 2k$$

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:

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

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

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 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 $\sim U(0, A_v)$. In the same manner as on-cast, the n^{th} iteration of on-NPC multiplicative boosts can be can be calculated as:

$$d_n = d_{n-1} + \lfloor d_{n-1} \cdot x_n \rfloor \tag{11}$$

Table 2: 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
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Hex hunter	DPL	Melee bane weapons	Vulnerability
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Greater sonic wave	Pernix quiver	Slayer helm	Smoke Cloud
	$\downarrow$	$\downarrow$	$\downarrow$
	Additive berserk effects	Fort Forinthry guardhouse	Gloves of Passage (bleed)
	$\downarrow$	$\downarrow$	$\downarrow$
	Prayer	Genocidal	X Slayer Perk
	$\downarrow$	$\downarrow$	$\downarrow$
	Damage boosting ultimates	Salve amulet	X Slayer Sigil
	$\downarrow$	$\downarrow$	$\downarrow$
	Exsanguinate	Ripper claw	Aura boost & Metamorphosis
	$\downarrow$	$\downarrow$	$\downarrow$
	Revenge	Ripper passive	Scrimshaws
	$\downarrow$	$\downarrow$	
	Spendthrift	Precise	
	$\downarrow$	$\downarrow$	
	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. Weapon damage (W_d) is used to randomly determine the damage dealt by auto attacks where $d \sim U(0, W_d)$. Auto attacks are subject to damage effect modifiers in the same manner as abilities, therefore the final minimum hit is rarely ever actually zero.

For any of the triangle combat styles, auto attacks can be forced by clicking the target while casting an ability immediately following non-damaging abilities. For example, following defensive abilities such as freedom and anticipate, or damage boosting ultimates, like sunshine, death swiftness, and berserk. Players have also discovered a more lucrative way to force these auto attacks by taking advantage of different attack frequencies.

The rate at which auto attacks fire is denoted on the tool-tip for the weapon as "attack rate". Two-hand weapons tend to have higher weapon damage but have a slower attack frequency (six ticks). Meanwhile main-hand and off-hand weapons have a four tick attack rate. The auto frequency is checked and updated based on the worn weapon after each ability cast. Therefore, a player using magic can cast a two hand auto while casting an ability before switching to dual wield weapons for the next ability cast then switch back to a two hand weapon and cast an auto four ticks after the dual wield ability cast.

Magic is the only style capable of taking advantage of this mechanism because it is the only style where auto attacks can be cast by player input. Nonetheless, it has become an integral part of any magic damage rotation sacrificing a tick on certain ability casts to weave in auto attacks which players have called "four tick auto-attacking", "4TAA", or "four-ticking." The name is derived from the fact that abilities cast with autos are the on the fourth tick rather than the third tick after a GCD. A common 4TAA structure would be *Auto + ability* $\rightarrow 3_t \rightarrow$ *dual-wield ability* $\rightarrow 4_t \rightarrow$ *Auto + ability.*

It should be mentioned that the auto cool downs were originally determined by the last equipped weapon on a given game tick rather than with an ability cast. This allowed a method that has since been removed called "continuous four ticking", or "C4TAA." The players would cast a two hand auto before switching to dual wield weapons for the ability cast all in the same game tick. This switching allowed the player to get an extra auto on the fourth tick of every GCD as opposed to every other GCD. The outcome of this method is that players while using magic would continuously cast abilities on the fourth tick to get the extra auto attack–rather than the three tick GCD–hence the name continuous four ticking.

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 of 12k instead of the normal 10k, but can be increased to 15k with an Erethdor's Grimoire. There are also various effects that can increase critical strike damage. Forced critical hits are rolled before the natural damage roll and the forced critical hit chance is determined by the sum of all critical hit chance boosting effects. When a forced critical strike occurs the variable damage range of the ability is set to the interval

$$d_f \sim U[\lfloor 0.95 \cdot A_v \rfloor, A_v]$$

Rather than using A_f as the minimum bound of the roll. The other critical hit mechanism are natural critical hits that occur when the player fails to get a forced critical hit, but the damage roll (d_r) is such that

$$d_r \geq \lfloor 0.95 \cdot (A_m) \rfloor$$

The hit is then subject to the same critical hit cap and critical damage boosts as forced critical strikes.

Necromancy critical strikes are mechanically different than 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:

$$d_f = \lfloor d_r \cdot C_d \rfloor$$

where d_r is the value determined by the damage roll and C_d is the associated critical hit damage modifier for the player's necromancy level. This system has a significant impact on the shape of the damage distribution of necromancy abilities further discussed in section 8.

7 Ability Oddities

7.1 Corruption 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 DPS 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 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 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 and for even n is the southwest tile of the 2×2 grid in the center of the matrix. If A_n is an $n \times n$ matrix the centers for $n = 5$ and $n = 4$ are as follows:

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

The center of the crystal rain 5×5 matrix is then randomly determined within an area defined by the rounded down radius of the original npc matrix around the center, using the same example, 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.

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

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.

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

When an arrow lands on a tile in the crystal rain matrix, higher-order arrows can no longer land on that tile. If an arrow rolls a tile that is already occupied by a previous arrow, it will re-roll the tile up to ten times. 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

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

Meaning, auxiliary arrows from multi-source SGB casts that are 8 or higher have zero variable damage[3].

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

$$\begin{aligned} A_f &= \lfloor 0.2 \cdot (A_d + l_d + s) \rfloor \\ A_v &= \lfloor 0.8 \cdot (A_d + l_d + s) \rfloor \end{aligned}$$

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 have zero variable damage, they deal the same damage every cast with the application of on-cast and on-npc effects.

7.6 Greater Ricochet

7.7 Snapshot

The variable damage of the second hit of Snap shot is calculated as scaled damage of the hit one variable damage[4].

$$\begin{aligned} f_2 &= f_1 \\ v_2 &= \lfloor 1.1 \cdot v_1 \rfloor \end{aligned}$$

7.8 Hurricane

For the second hit of hurricane DPL is calculated as

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

[3]Sfox. "Seren Godbow", *Explanation of Arrow Damages*, The RuneScape Wiki.
[4]Sfox. "Snapshot", *Explanation of Arrow Damages*, The RuneScape Wiki.

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[5].

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[6].

7.10 Bleeds

Combust, fragmentation shot, dismember, and slaughter all roll a value between 1% and $max\%$ 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 on fixed damage, 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, providing clarity on 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 offer a comprehensive understanding of 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

[5]Sfox. "Hurricane", *Explanation of Hurricane Hits*, The RuneScape Wiki.

[6]RSN Veggie discovered the 25% min hit for the hit derived PE proc damage which was then confirmed in the code by Mod Sponge

critical hits. The weighted mean μ_a of an ability a is the expected value given by:

$$\mu_a = E[a] = \sum_{d \in a} d \cdot P(d) \tag{12}$$

where d represents a possible damage value in the domain of a and $P(d)$ 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 (14). 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 is 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 is 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 an ability are expected to deviate from the average damage μ_a. For an ability a, the variance is calculated as:

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

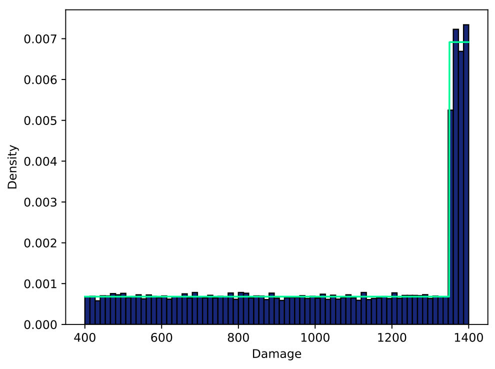
where d is a possible damage value, μ_a is the weighted mean of the damage, and $P(d)$ is the probability of that damage occuring. The standard deviation σ is the square root of the variance $\sqrt{\sigma^2}$ and provides a more intuitive measure of variability because it is in the same units as the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

8.3 Distributions of Single-Hit Abilities

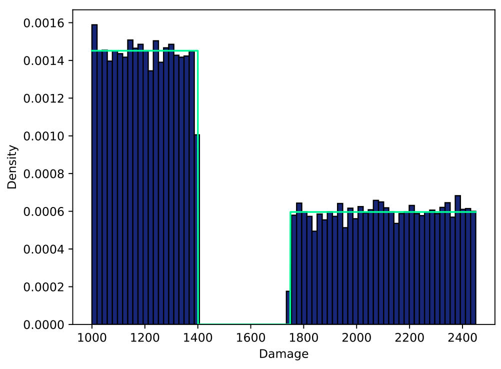
The damage roll of a single hit ability is discrete and uniform; any damage value in a particular domain is equally likely to occur. The most appropriate way to present this mathematically is to say that the damage of an ability can exist within two domains–λ_r for a natural roll and λ_f for a forced 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 likely hood is the sum of the two probabilities. Take λ_n as the vector of possible natural rolls and λ_f as the vector of possible forced crits, the probability mass function (PMF) of an ability $p(a)$ is defined as:

$$p(a) = \begin{cases} P(r) \cdot \frac{1}{r_n} \quad a \in \lambda_n \text{ and } a \notin \lambda_f \\ P(r) \cdot \frac{1}{r_n} + P(f) \cdot \frac{1}{f_n} \quad a \in \{\lambda_n, \lambda_f\} \\ P(f) \cdot \frac{1}{f_n} \quad a \notin \lambda_n \text{ and } a \in \lambda_f \\ 0 \quad a \notin \{\lambda_n, \lambda_f\} \end{cases} \tag{13}$$

The distribution produced by the PMF in equation (15) 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 4.



(a) Combat triangle styles



(b) Necromancy

Figure 4: $p(a)$ 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, standard deviation is the square root of the sum of individual variances, and the PMF φ_a is the convolution of the individual PMFs[7]. 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:

$$p(a) = p_x(x) * p_y(y) = \sum_{k=-\infty}^{\infty} p_x(k)p_y(a-k) \tag{14}$$

Instead of performing convolution on the PMFs for each hit, we can evaluate for the probability vectors produced by the PMFs p_x and p_y.

$$p_x = [p_1, p_2, p_3, \cdots, p_n]$$

$$p_y = [p_1, p_2, p_3, \cdots, p_n]$$

Utilizing the Convolution theorem of the Fourier Transform, the Discrete Fourier Transform (DFT) for our vectors is defined as:

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

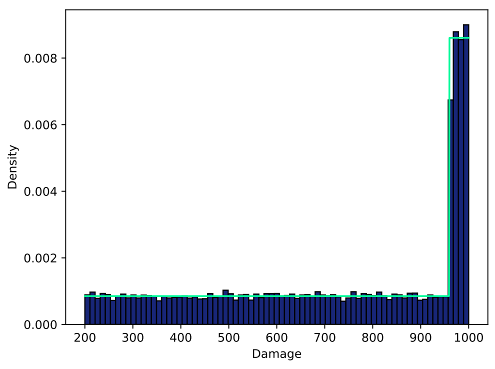
$$\hat{p}_y(\xi) = \sum_{n=0}^{N-1} p_y e^{-i(\frac{2\pi}{N})\xi p_n}$$

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 $\hat{p}_a$

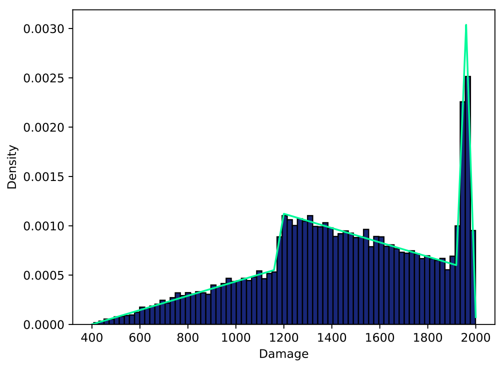
$$p_a = \frac{1}{N} \sum_{k=0}^{N-1} \hat{p}_a(\xi) e^{i(\frac{2\pi}{N}\xi p_n)}$$

[7]Grinshpan, Anatolii. "Discrete Convolution", Drexel University.

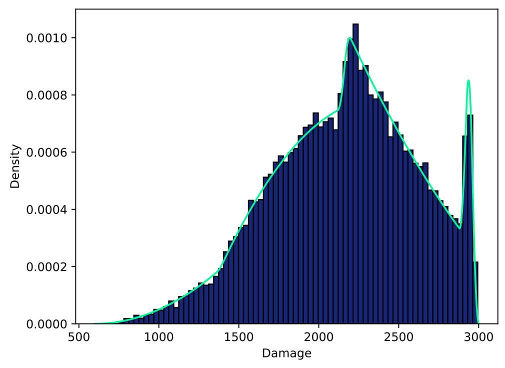
The resulting vector p_a is the probabilities of the damage vector λ_a for a two hit ability. The probability vector for a k hit ability is calculated by recursively performing these Fourier transforms for the resulting vector and the next hit.



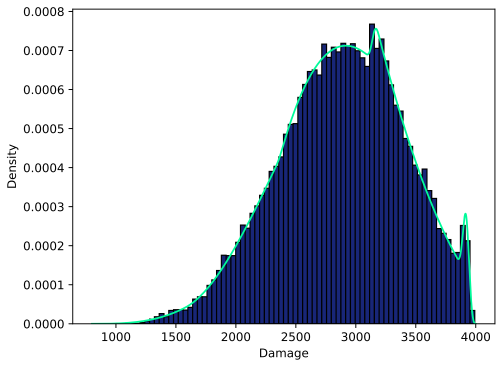
(a) 1-Hit



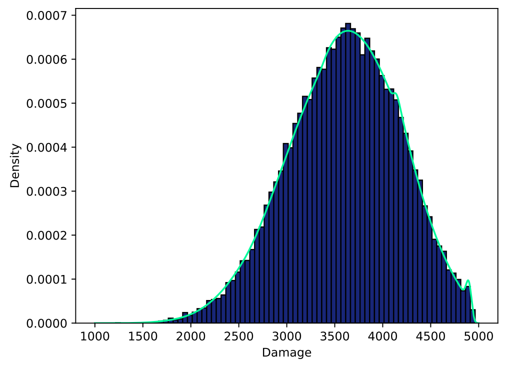
(b) 2-Hit



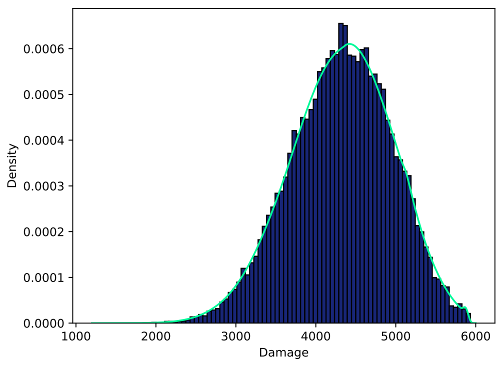
(c) 3-Hit



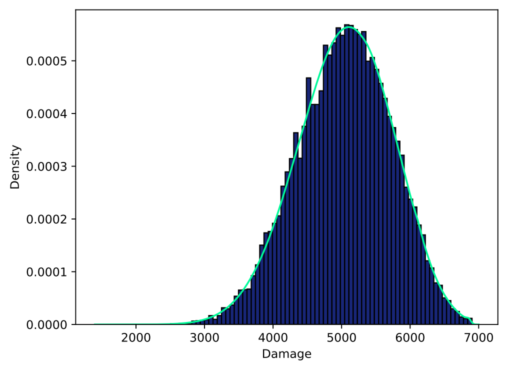
(d) 4-Hit



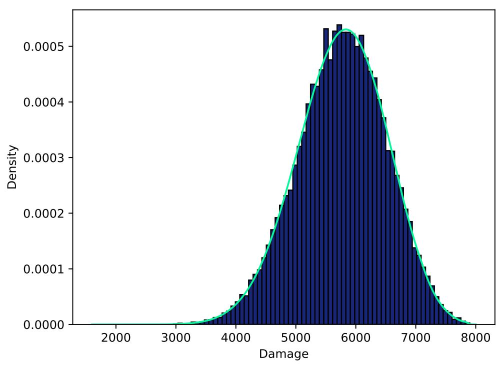
(e) 5-Hit



(f) 6-Hit



(g) 7-Hit



(h) 8-Hit

Figure 5: 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 that 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 length of the rotation increases. A viable alternative when conducting analysis on 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 length of the rotation becomes infinitely long. Suppose r is a rotation made up 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 we expect the convergence of r to be $N(0,1)$. A common proof of classical CLT utilizes the characteristic function for the ability defined as the Fourier transform its PMF.

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

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

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

Showing 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, therefore in most circumstances we are looking at non-identically distributed variables. A requisite parameter of classical CLT is that the independent variables are identically distributed, however, we find that most non-identically distributed sets of abilities in RuneScape converge in a similar manner. 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 $\delta > 0$

$$\lim_{n\to\infty} \frac{1}{\sigma_r^{2+\delta}} \sum_{j=1}^{n} E\left[\left|a_j - \mu_{a_j}\right|^{2+\delta}\right] = 0 \tag{15}$$

where $E\left[\left|a_j - \mu_{a_j}\right|^k\right]$ gives the k^{th} raw moment of the distribution. 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 behaviour 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. We utilize the moment generating function(MGF) of r where

$$\psi_r(t) = E[e^{tr}] = \sum_{r_i} e^{tr_i} p_r(r_i) \tag{16}$$

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

$$\gamma_r = \dddot{\psi}_r(t)$$
$$\kappa_r = \ddddot{\psi}_r(t)$$

The raw moment for kurtosis in a true Gaussian is three which is standardized by simply subtracting three to obtain excess kurtosis (standardized moment) k' of zero. As an example, let us calculate the convergence of a rotation r when it is comprised of *Greater conc.*$\rightarrow$ *Wild magic*$\rightarrow$ *Auto + 3-hit Asphyxiate* where $A_d = 2000$, $W_d = 1000$, and $P(f) = 0.318$. Note that greater conc increases crit 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

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

values in Table 2, we can evaluate the convergence of γ_r and 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

Table 4: Convergence of γ and κ'

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

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 6.

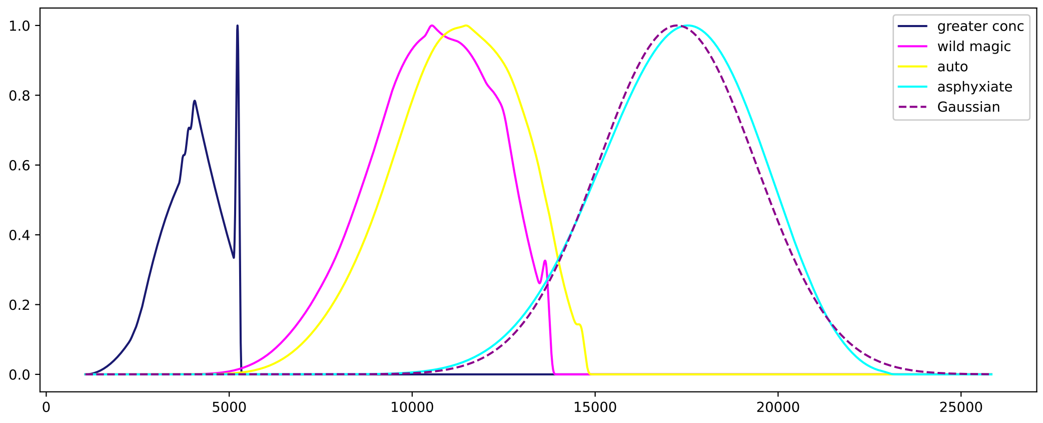


Figure 6: Visual convergence of $p_r(d)$ to a Gaussian $p_g(d)$ with each added ability. Convergence is highly variable, for example, necromancy style abilities with bi-modal distributions take much longer to converge than trio styles. The convergence should be evaluated for the specific rotation in question rather than universally applied after n hits.

10 Comparative Rotation Analysis

Rotations are rarely ever evaluated in their entirety because there tend to be transition points throughout the fight that disrupt the flow of actions. Nested rotations are smaller sequences that exist within rotations and are split up by local endpoints. The rotation then is the collocation of these independently formulated nested rotations. For example, hard-mode(HM) Vorago, the fight is 11 phases and a nested rotation in this context is the sequence for one of those phases. The moment that Vorago's life points are depleted on a given phase, marking the end of the phase, would be a local endpoint.

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 they want to know how likely they are to hit some damage threshold we calculate it using a continuous cumulative distribution function. 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 φ_x

$$p_x(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{x-\mu_x}{2\sigma}\right)^2} \tag{17}$$

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

$$P(x \geq n) = 1 - \Phi_x(x)$$

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

$$\Phi_x(x) = \int_{-\infty}^{x} p_x(t)dt \tag{18}$$

Using these methods, we can also analyze the comparative damage output of two rotations, r_x and r_y. The probability that $x > y$ is determined by the difference between their respective Gaussian distributions. Let λ_x and λ_y be the damage vectors for rotations r_x and r_y, with means μ_x, μ_y and standard deviations σ_x, σ_y, respectively. The random variable $z = x - y$ represents the damage difference between the two rotations. Since r_x and r_y are assumed to be independent and Gaussian, 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 x exceeds y ($z > 0$) can be computed using the cumulative distribution function (CDF) of z:

$$P(z > 0) = 1 - \Phi_z(0)$$

where $\Phi_z(0)$ is the CDF of z evaluated at 0, given by

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

Evaluating the integral yields the total probability that x is greater than y.

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 $\lambda_x = x_1, x_2, \ldots$ with probability $p(x_i)$. The CDF then would be

$$\Phi_x(x) = P(x \geq n) = \sum_{x_i \geq n} p(x_i) \tag{19}$$

Similar to the continuous case, the probability of one rotation out damaging another is given by the difference $z = x - y$. Which is evaluated for the discrete CDF Φ_z for $z \geq 0$. Comparing the raw damage distributions for rotations is most useful when they have similar time intervals. For more nuanced analysis, alternative measurements can be considered such as damage per tick (dividing damage by the rotation length in ticks) or damage per adrenaline.

11 The Dynamic Programming Approach

The rotation analysis in section 10 details how we can quantitatively asses the comparative damage output of different rotations. However, we do not know that either of the options are the most optimal set of actions. With some basic heuristics we can make some improvements but these improvements do not give us any empirical proof that a rotation is truly optimal. If we desire the true optimal sequence of actions across an encounter we need to utilize principles of dynamic programming.

11.1 The Bellman Equation for Deterministic Sequences

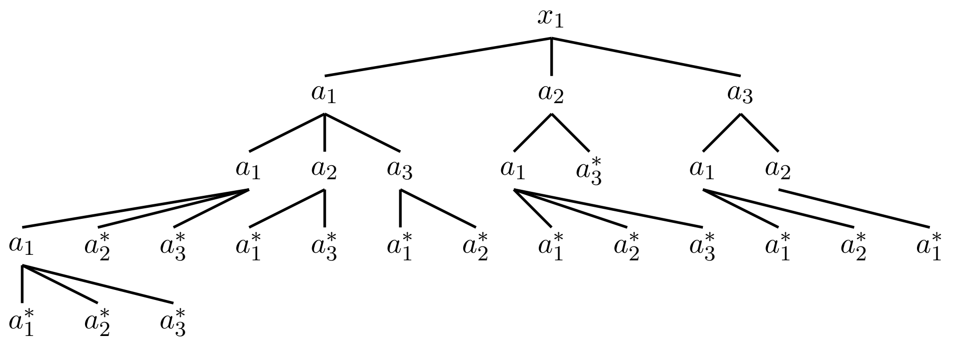
Finding the optimal sequence of actions starts with a brute force approach where a given state x_t is solely determined by the action a_t of the x_{t-1} state. Let us start with a simple deterministic landscape where we have three abilities $[a_1, a_2, a_3]$ with the following parameters

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

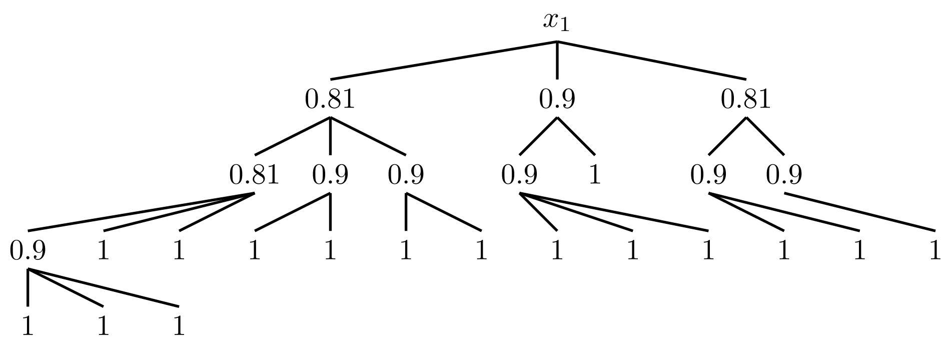
The cooldown denotes the number of states that must pass before an ability can be cast again. The Bellman equation helps us evaluate the utility of each potential action relative to the advantage or disadvantage of each state.

$$V(x_t) = \max_a(U(x_t, a_t) + \beta V(x')) \tag{20}$$

Where $U(x_t, a_t)$ 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 backwards induction to compute the utility gained from for each action in the state x_t. If the terminal state is obtained when the sequence dealt exceeds 50 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 then would be $a_2 \to a_3 \to a_2$ 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 is called value iteration, using the value function to iterate over all states and extract the optimal rotation. RuneScape combat is much more complex than this, there are many probabilistic elements to consider that 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 principle challenge of solving an MDP for a RuneScape rotation is the exponentially large number of possible rotations. Even if we assume no cool down dynamics then the number of possible rotations r_n is $r_n = a_n^{x_n}$. For example, a world with 4 actions and 10 states has 4^{10} possible rotations. In RuneScape, rotations can be on the order of hundreds of actions with thousands of possible states. 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. Policy in the context of RuneScape would be not only the ability rotation but every action the player takes tick by tick for the duration of the fight.

11.2.1 Policy Iteration

In order to carry out policy iteration we first much 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 of all possible state sequences defined by the policy.

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

$V^\pi(x_t) = 1$ for terminal states, meaning the probability of the boss 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, $Q^\pi(x_t, a)$ that yields the expected value of an action a–as opposed to the policy action–and then following the policy π.

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

If there exists an action a where $Q^\pi(x_t, a) > Q^\pi(x_t, \pi(x_t))$ then the policy is strictly improved by $\pi(x_t) \leftarrow a$. 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 $x_t \in X$, adjust π such that

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

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

11.2.2 Policy Gradients with Actor Critic Approximations

11.2.3 Implementation

12 Conclusions

12.1 Impending changes