CallCulator Handbook

Advanced version

Aaron Gonsior

February 25, 2026

Contents

Part I
Foundations

1 Welcome to CallCulator

CallCulator is an interactive web-based tool for quantitative options analysis. It enables users to move beyond zero-dimensional metrics (price targets) and instead express their market beliefs through probability distributions, which are then combined with risk tolerance profiles to find mathematically optimal investment strategies.

The platform offers three main modes, accessible via the tabs at the top of the page:

1. **Basic Calculation** – Manually define a custom option spread and compute its full performance profile, Black-Scholes time-decay surface, and Monte-Carlo CAGR statistics.
2. **Probability & Risk** – Supply a probability distribution and risk tolerance to let the backend search engine find the globally optimal strategy across hundreds of millions of option combinations.
3. **Max Min** – Find strategies that maximize the worst-case outcome (minimax optimization), either for a single stock or across a multi-stock portfolio.

2 Basics of Options Trading

2.1 What is an Option?

An **option** is a financial derivative that grants the holder the right—but not the obligation—to buy or sell an underlying asset at a pre-agreed price (the **strike price** K) on or before a specified date (the **expiration date** T).

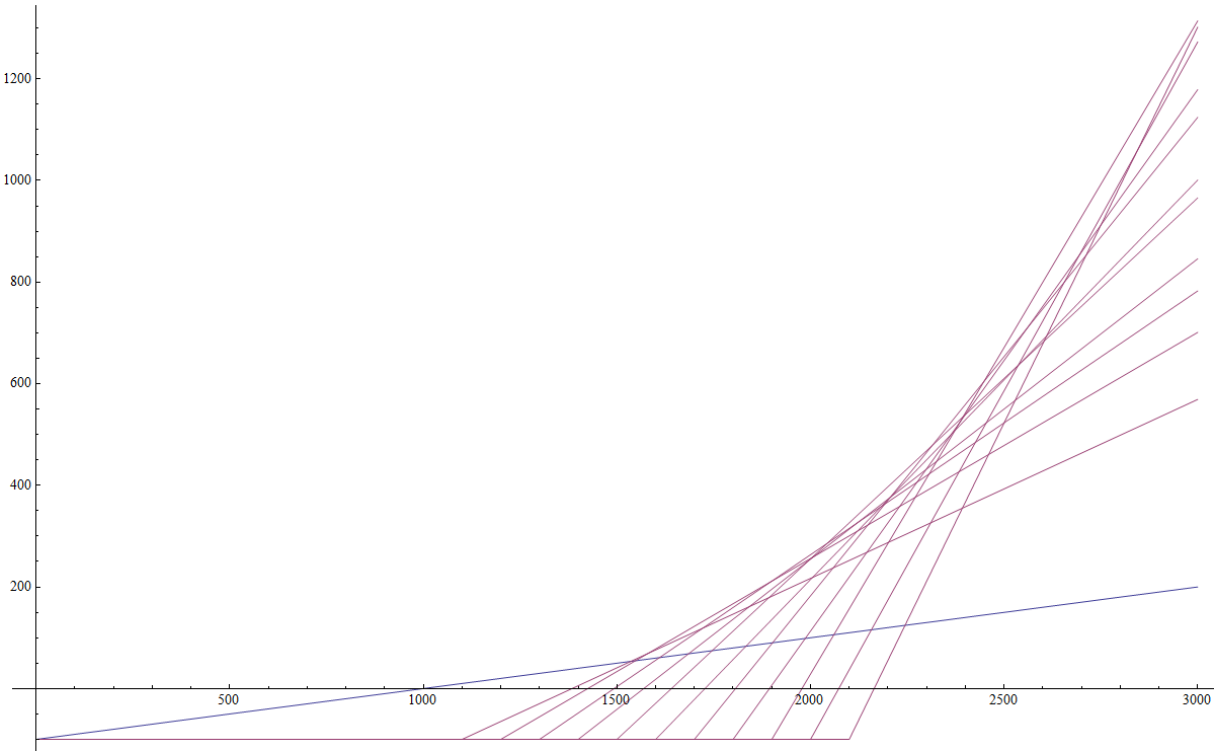
- **Call Option** $C(K,T)$: Right to *buy* at K. Payoff at expiration: $\max(S_T - K, 0)$ where S_T is the stock price at time T.
- **Put Option** $P(K,T)$: Right to *sell* at K. Payoff at expiration: $\max(K - S_T, 0)$.

The **premium** p is the market price of the option. In percentage terms, the profit/loss of a call option is:

$$g_c(S_T) = \max\Big(-1,\ \frac{S_T - K - p}{p}\Big) \tag{1}$$

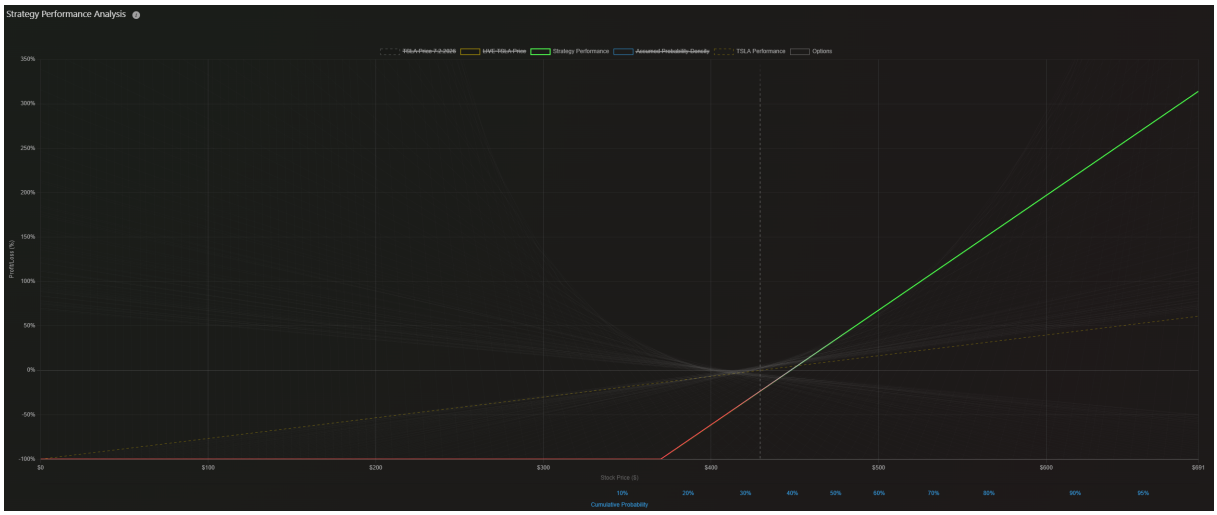
2.2 Visualizing Option Performance

The graph below shows the percentage gain or loss of different call options (red lines) compared to simply owning the stock (blue line):



Red: percentage gain of various call options. Blue: the underlying stock.

Notice how the red lines are much steeper than the blue line. A 10% rise in the stock price might translate to a 50% or even 200% rise in a call option's value. But if the stock doesn't reach the strike price, the option goes to zero—a 100% loss. This is the leverage effect in action.



A single call option's performance at expiration.

2.3 American vs. European Options

While you can trade (buy/sell) any option on the market at any time, there is a distinction in when you can **exercise** the right to buy or sell the underlying stock:

- **American-Style:** You can exercise the option at any point up to and including the expiration date. Most standard equity options are American-style.
- **European-Style:** You can only exercise the option on the expiration date itself. Index options are often European-style.

For most traders, this distinction is technical because options are rarely exercised early; they are usually sold to close the position.

2.4 What is a Spread?

A **spread** is a convex combination of financial products:

$$\sum_i w_i \, g_i \quad \text{where} \quad \sum_i w_i = 1, \; w_i \in [0, 1] \tag{2}$$

Spreads allow you to engineer a specific risk/reward profile. CallCulator supports multi-leg strategies including **short (write) positions**, enabling bull spreads, bear spreads, iron condors, and custom combinations. The search algorithms don't know any difference between those traditionally grouped spreads into names, it is completely arbitrary.

Part II
Core Philosophy: Why Probability Distributions?

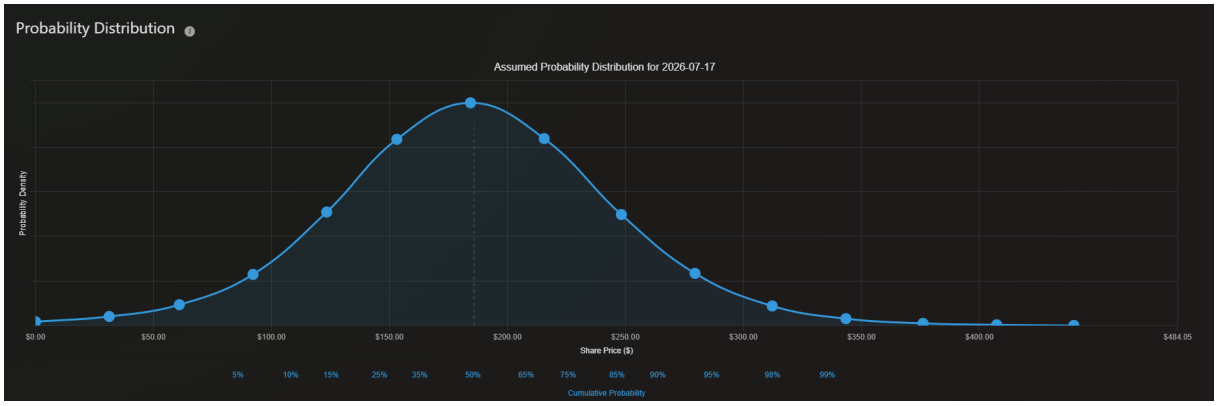
3 The Problem with Price Targets

Conventional analysis relies on a "price target"—a single number X representing where you think a stock will be by some date. This is a zero-dimensional metric that collapses all uncertainty into a point estimate.

The problem: two investors may both have a target of \$150, yet one might be very confident (narrow bell curve around \$150) while the other sees wild uncertainty (flat distribution from \$80 to \$220). A price target cannot distinguish these beliefs, leading to fundamentally different appropriate strategies.

4 Our Approach: Probability Density Functions

CallCulator replaces the price target with a **Probability Density Function (PDF)** $\mu(x)$ for the stock price at a future date. This captures the full shape of uncertainty:

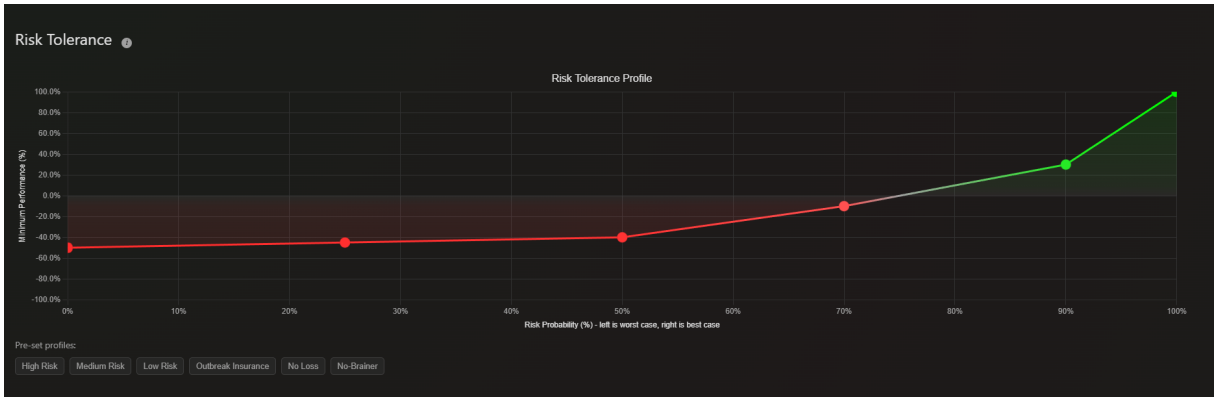


A probability distribution supplied by the user: the peak at $\sim$ \$180 indicates the most likely price.

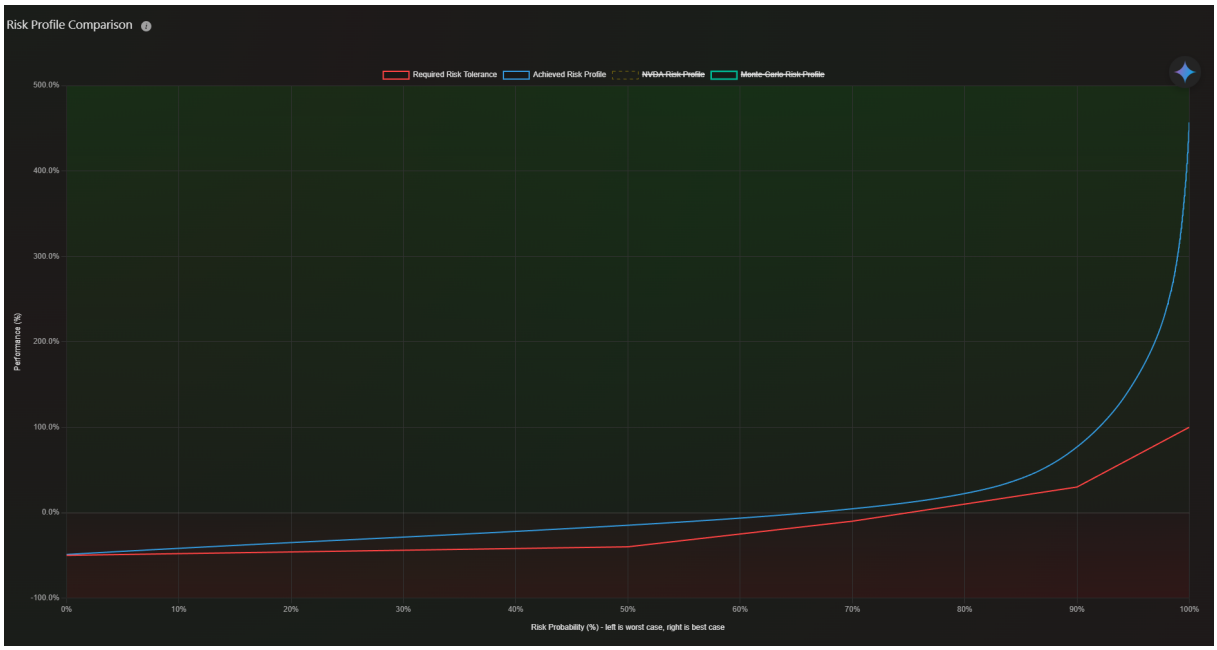
Given $\mu(x)$ and a set of chosen options $\{g_i\}$, the **expected return** of a strategy is:

$$\mathbb{E}_\mu\Big[\sum_i w_i \, g_i\Big] = \sum_i w_i \int_{\mathbb{R}} g_i(x) \, \mu(x) \, dx \tag{3}$$

Combined with a **Risk Tolerance Profile**, we can find the strategy that maximizes expected return subject to the constraint that its risk profile stays within the user's tolerance at every cumulative probability level.



The user's risk tolerance profile $\hat{r}_{tol}(\lambda)$: a continuous boundary from worst case (left) to best case (right).



The found strategy's risk profile $\hat{r}_c(\lambda)$ (blue) satisfying $\hat{r}_c(\lambda) \geq \hat{r}_{tol}(\lambda)\ \forall \lambda \in [0,1]$

4.1 From PDF to Risk Profile: The Mathematical Bridge

The probability distribution $\mu(x)$ and a strategy's performance function $p_c(x)$ together determine a **risk profile** $\hat{r}_c : [0,1] \to \mathbb{R}$—a non-decreasing curve that maps each cumulative probability level λ to the return achieved at that quantile. This is the object plotted in the Risk Profile Comparison charts.

Monotone strategies (single legs). For a single option whose performance $p_c(x)$ is monotone in the share price, the construction is direct. Define the CDF $\Psi(x) = \int_0^x \mu(y)\,dy$, which maps prices to cumulative probabilities. The risk profile is:

$$\hat{r}_c(\lambda) = p_c\big(\Psi^{-1}(\lambda)\big) \tag{4}$$

The CDF inverse $\Psi^{-1}(\lambda)$ converts the probability level λ to the corresponding share price ("at probability level 30%, the stock is at \$X"), and p_c converts that price to the strategy's return. Since both Ψ^{-1} and p_c are monotone, $\hat{r}_c$ is itself non-decreasing.

Non-monotone strategies (spreads). For multi-leg spreads, $p_c(x)$ is generally not monotone—it may rise, peak, and fall. In this case, different share prices can produce the same return. The risk profile is then the **quantile function of the induced return distribution**:

$$\hat{r}_c(\lambda) = \inf\Big\{ z \in \mathbb{R} \ : \ \int_{\{x\,:\,p_c(x)\le z\}} \mu(x)\,dx \ge \lambda \Big\} \tag{5}$$

The integral computes, for each return level z, the total probability of achieving a return *at most* z—integrating over all (possibly disjoint) price regions where the strategy underperforms z. Inverting this CDF produces the non-decreasing risk profile.

In practice, the computation proceeds as follows: First, the multi-leg spread is converted into a performance curve $p_c(x)$ by combining all option payoffs with their respective weights. A series of return levels z is sampled uniformly across the range from the minimum to maximum possible performance. For each return level, we identify all price intervals where the strategy's payoff does not exceed z—these are the sub-level sets $\{x : p_c(x) \le z\}$, which may consist of multiple disjoint regions when the performance curve is non-monotone. We then integrate the probability density $\mu(x)$ over all these intervals using polynomial spline arithmetic, accumulating the cumulative probability for this return level. Finally, these (return, probability) pairs are inverted and fit to a spline to construct the risk profile.

The optimization constraint. The risk tolerance profile $\hat{r}_{tol}(\lambda)$ is the user's minimum acceptable return at each quantile. The optimization engine enforces:

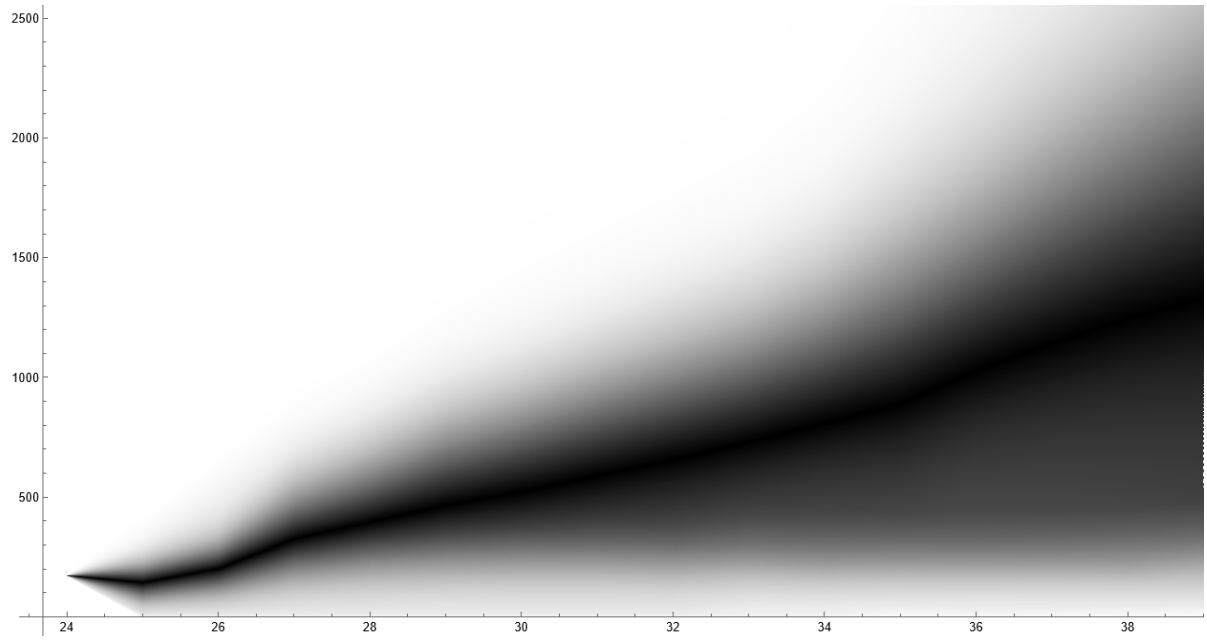
$$\hat{r}_c(\lambda) \ge \hat{r}_{tol}(\lambda) \quad \forall\, \lambda \in [0,1]$$

This ensures that at *every* probability level—from the worst tail to the best—the strategy meets or exceeds the user's requirements.

5 Probability Distributions Over Time
(In Development)

For dynamic strategies, we model beliefs as a family of distributions $\{\mu_i(x)\}_{i\in[n]}$ at times $\{d_i\}$, then interpolate continuously using linear transport:

$$\nu(x,d) = \frac{d_{i+1}-d}{d_{i+1}-d_i}\,\mu_i(x) + \frac{d-d_i}{d_{i+1}-d_i}\,\mu_{i+1}(x), \quad d \in [d_i, d_{i+1}] \tag{6}$$



Grayscale heatmap of a probability distribution evolving over time.
(This is my personal belief of $TSLA share price development as per 2024)

Part III
Using CallCulator

6 Basic Calculation Mode

6.1 Workflow

1. **Enter a Ticker:** Type a stock symbol (e.g., `TSLA`) and press Enter. The system fetches the current price, company name, FIGI code, and full options chain via the backend Data Manager.
2. **Select an Expiration Date:** Choose from the list of available expirations.
3. **Build Your Strategy:** Add legs (Buy Call, Buy Put, Sell Call, Sell Put) with strike prices and weights (percentages representing capital allocation, e.g., 60% + 40%).
4. **Compute:** The backend `BasicCalcs` Lambda calculates the full analysis.

6.2 Results Displayed

- **Strategy Performance Chart:** A graph showing your percentage gain/loss (y-axis) at every possible stock price (x-axis) at expiration.
- **Black-Scholes Table:** A grid of estimated option values across time (columns) and stock price (rows), using the Black-Scholes PDE solution. Color-coded: green = profit, red = loss.

  The table axes are:

  - **X-axis (Time):** Time progresses from left (today) to right (expiration date).
  - **Y-axis (Price):** Possible share prices of the underlying stock.

  The cell at coordinate (x, y) shows the estimated P/L of your strategy at that specific future time and stock price. The rightmost column (at expiration) corresponds exactly to the values in the Strategy Performance Chart above it (just rotated 90 degrees). Values at expiration are certain (intrinsic value), while values earlier in time are theoretical estimates based on the Black-Scholes model.
- **Monte-Carlo CAGR Statistics:** Simulated compound annual growth rate assuming repeated investment with the given probability distribution.
- **Action Buttons:** Share (generates public link), Track, Invest (in development).

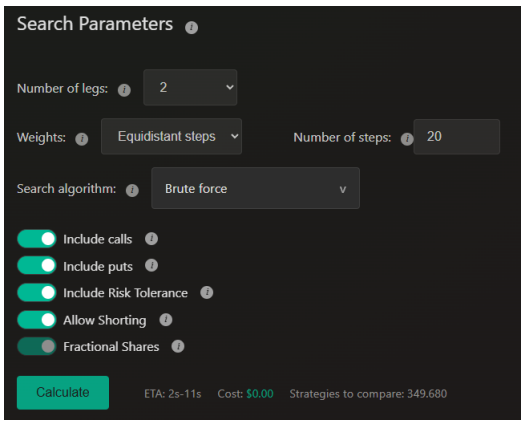
Note on Black-Scholes realism: The standard Black-Scholes model assumes constant implied volatility (IV) per option leg. In reality, IV changes as the stock price moves. CallCulator offers a **Dynamic IV** mode (enabled via a toggle in Search Parameters) that models this effect. See Section 10.2 for details.

7 Probability & Risk Mode

This is CallCulator's flagship mode. Instead of manually defining a strategy, you define your *beliefs* and let the optimization engine find the best strategy.

7.1 Workflow

1. **Enter a Ticker:** Same as Basic mode.
2. **Select an Expiration Date:** The system pre-fetches all implied PDFs in a single backend call (`/impliedPDFAll`).
3. **Define Your Probability Distribution:** Adjust the probability curve $\mu(x)$ to match your belief about where the stock will be at expiration (see figure below).
4. **Set Risk Tolerance:** Define a continuous tolerance function $\hat{r}_{tol}(\lambda)$ mapping each cumulative probability λ to the minimum acceptable return at that level (e.g., "in the worst 40% of outcomes, I require at least -5%").
5. **Configure Search Parameters:** Choose the number of **legs** (option positions per strategy), the number of **steps** (granularity of the strike/weight grid), and the search algorithm. Currently a **Brute Force** scan is used; smarter algorithms (Quasi-Monte Carlo / Sobol seeding, adaptive hill-climbing) are in development. Complexity grows rapidly with legs and steps, affecting computation time and cost.



Search parameter configuration panel.

6. **Launch Search:** The backend Search Manager distributes the optimization across multiple Lambda worker nodes, each evaluating hundreds of millions of option combinations in parallel (up to 625 threads per node, $\sim$80 calculations per thread).

7.2 Search Algorithms

CallCulator employs two distinct algorithmic approaches to solve the non-convex optimization problem of finding the best spread.

7.2.1 Brute Force

The Brute Force algorithm discretizes the search space (strike prices and weights) into a uniform grid and evaluates every single valid combination.

- **Pros:** Guaranteed to find the global optimum within the resolution of the grid.
- **Cons:** Exponential time complexity $\mathcal{O}(N^k)$ where k is the number of legs. Infeasible for $k > 3$.

7.2.2 Hybrid Search (GSGD)

The **Grid Search Gradient Descent (GSGD)** algorithm is a hybrid global-local optimizer designed for high-dimensional spaces (4+ legs). It operates in two phases:

1. **Global Exploration (Sobol Seeding):** Instead of a uniform grid, the algorithm generates initial starting points ("seeds") using **Sobol sequences**. These are quasi-random low-discrepancy sequences that cover the multi-dimensional search space more evenly than pseudo-random numbers, avoiding clustering and gaps.
2. **Local Exploitation (Gradient Descent):** From each seed, the algorithm performs a gradient descent optimization to "climb the hill" of expected return. It uses **Powell-Wolfe step rules** to dynamically adjust the step size, ensuring that each update provides a sufficient decrease in the objective function (or increase in return) while satisfying curvature conditions.

This approach allows CallCulator to efficiently navigate the complex, multi-modal landscape of option strategies, finding high-quality solutions in spaces that are too vast for brute force.

7.2.3 Performance & Complexity

- **Brute Force:** Computational cost explodes exponentially with complexity ($\mathcal{O}(S^N)$). Great for 2-3 legs, impossible for 4+.
- **Hybrid Search (GSGD):** Computational cost scales linearly with complexity ($\mathcal{O}(K \cdot N)$). Feasible for 4-5+ legs. The runtime depends primarily on the number of seeds (K) chosen by the user, not the number of legs.
- **Mathematical Construction:** The theoretical "Insurer + Sniper" solution (Regime 2) is constructed analytically in $\mathcal{O}(M \log M)$ time by sorting the likelihood ratio. This is effectively instantaneous (milliseconds) regardless of strategy complexity, but produces an ideal continuous payoff that must be approximated by discrete options.

7.3 The Optimization Engine

The backend considers all 2- and 3-leg option spreads from the available chain. The optimization problem is formally defined as:

$$\max_{s \in \mathcal{S}} \mathbb{E}_P[g_s] = \max_{s \in \mathcal{S}} \int_0^\infty g_s(x) P(x)\, dx \tag{7}$$

Subject to the risk constraint:

$$\forall q \in [0,1] : R_s^{-1}(q) \geq T(q) \tag{8}$$

where $T(q)$ is the user's risk tolerance profile (minimum return at quantile q), and $R_s(y)$ is the cumulative distribution function of the strategy's return:

$$R_s(y) = \int_{\{x | g_s(x) \leq y\}} P(x)\, dx \tag{9}$$

Expanding the strategy s as a weighted sum of options $o_i \in \mathcal{O}$:

$$\max_{\substack{o_i \in \mathcal{O} \\ w_i \in \mathbb{R} \\ \sum w_i = 1}} \int_0^\infty \left(\sum_i w_i \cdot g_{o_i}(x) \right) P(x)\, dx \tag{10}$$

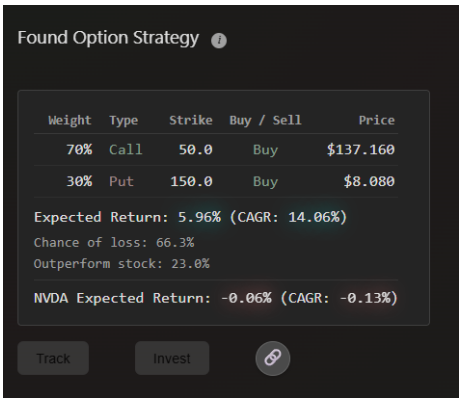
For each candidate:

1. Compute the performance function $g(S_T)$ using spline interpolation.
2. Compute the expected return: $\mathbb{E}_\mu[g] = \int g(x)\,\mu(x)\,dx$.
3. Compute the risk profile and verify it satisfies the risk tolerance constraint.
4. If valid, compare against the current best strategy.

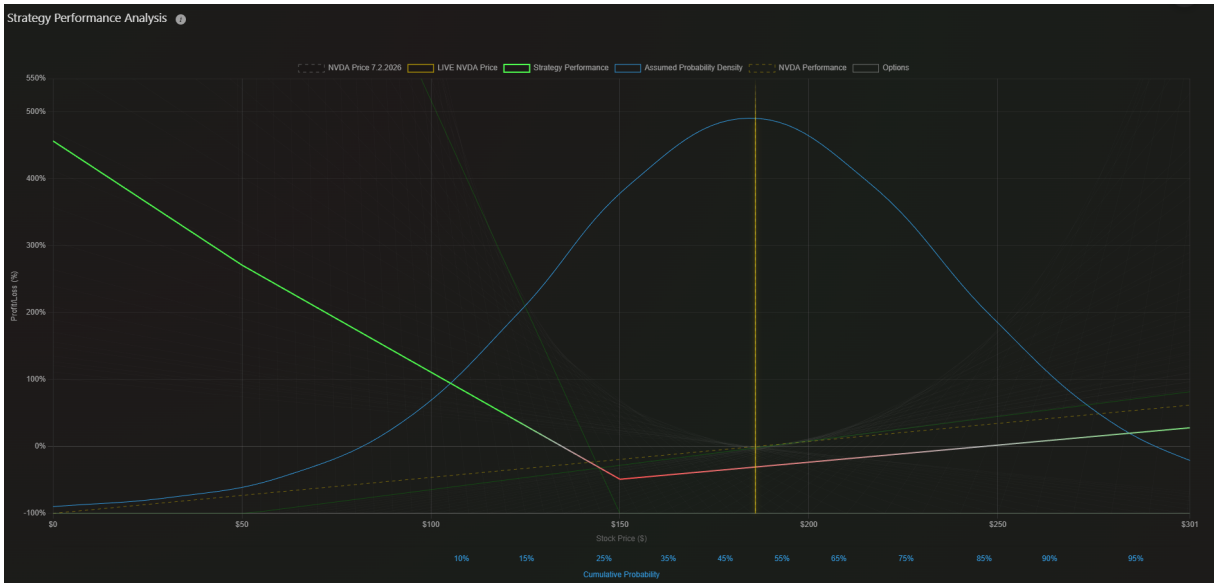
Weights represent **capital allocation** (percentage of portfolio). The backend derives option quantities as weight/option_price.

7.4 Results Displayed

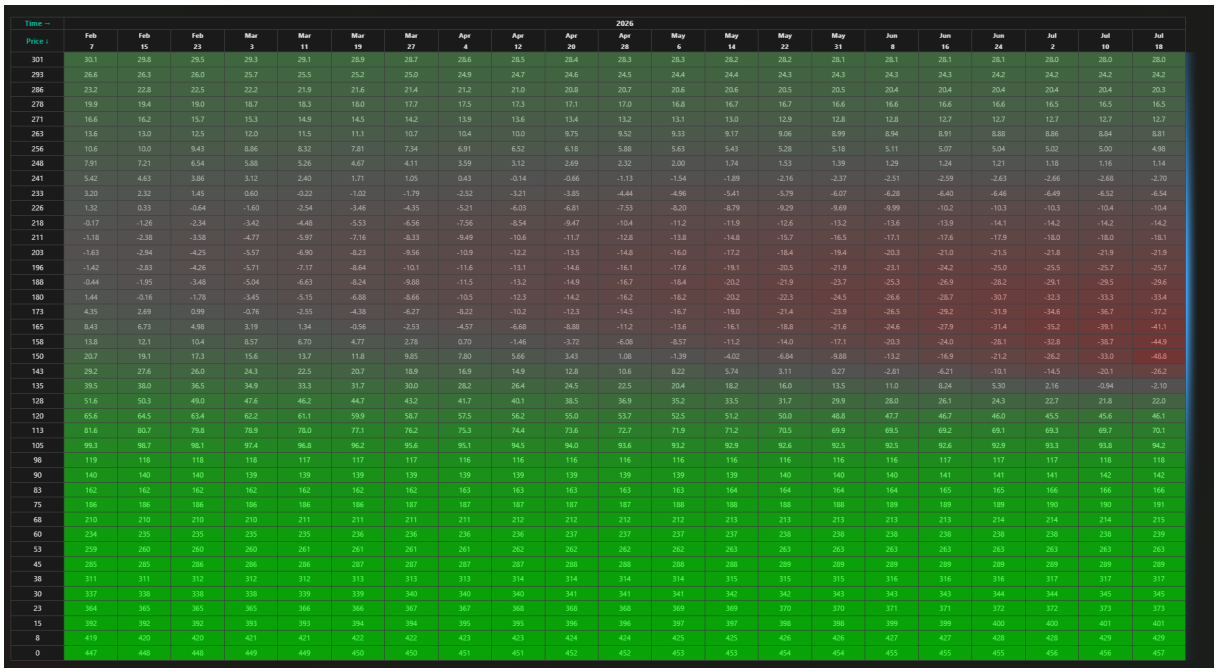
All results from Basic mode, plus:



Found Option Strategy card with legs, weights, expected return, and key statistics.



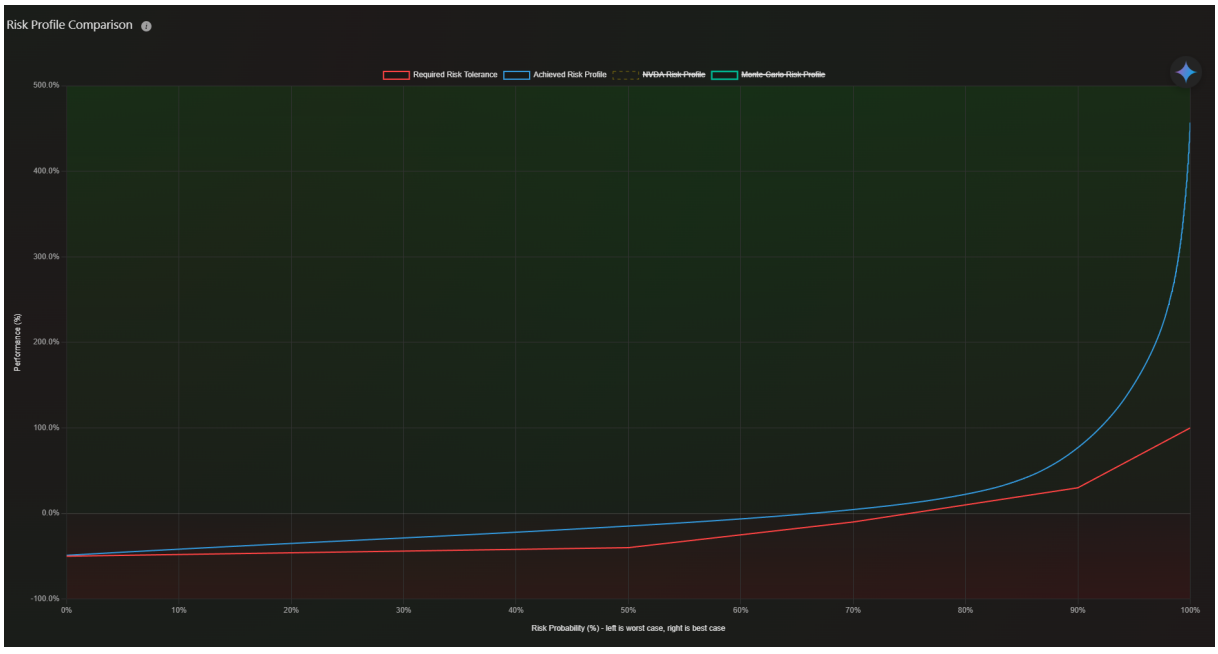
Strategy Performance Analysis: percentage return across all possible stock prices.



Time → / Price ↓	2026																			
	Feb 7	Feb 15	Feb 23	Mar 3	Mar 11	Mar 19	Mar 27	Apr 4	Apr 12	Apr 20	Apr 28	May 6	May 14	May 22	May 31	Jun 8	Jun 16	Jun 24	Jul 2	Jul 10	Jul 18
301	30.1	29.8	29.5	29.3	29.1	28.9	28.7	28.6	28.5	28.4	28.3	28.3	28.2	28.2	28.1	28.1	28.1	28.1	28.0	28.0	28.0
293	26.6	26.3	26.0	25.7	25.5	25.2	25.0	24.9	24.7	24.6	24.5	24.4	24.4	24.3	24.3	24.3	24.3	24.2	24.2	24.2	24.2
286	23.2	22.8	22.5	22.2	21.9	21.6	21.4	21.2	21.0	20.8	20.7	20.6	20.6	20.5	20.5	20.4	20.4	20.4	20.4	20.4	20.3
278	19.9	19.4	19.0	18.7	18.3	18.0	17.7	17.5	17.3	17.1	17.0	16.8	16.7	16.7	16.6	16.6	16.6	16.6	16.5	16.5	16.5
271	16.6	16.2	15.7	15.3	14.9	14.5	14.2	13.9	13.6	13.4	13.2	13.1	13.0	12.9	12.8	12.8	12.7	12.7	12.7	12.7	12.7
263	13.6	13.0	12.5	12.0	11.5	11.1	10.7	10.4	10.0	9.75	9.52	9.33	9.17	9.06	8.99	8.94	8.91	8.88	8.86	8.84	8.81
256	10.6	10.0	9.43	8.86	8.32	7.81	7.34	6.91	6.52	6.18	5.88	5.63	5.43	5.28	5.18	5.11	5.07	5.04	5.02	5.00	4.98
248	7.91	7.21	6.54	5.88	5.26	4.67	4.11	3.59	3.12	2.69	2.32	2.00	1.74	1.53	1.39	1.29	1.24	1.21	1.18	1.16	1.14
241	5.42	4.63	3.86	3.12	2.40	1.71	1.05	0.43	-0.14	-0.66	-1.13	-1.54	-1.89	-2.16	-2.37	-2.51	-2.59	-2.63	-2.66	-2.68	-2.70
233	3.20	2.32	1.45	0.60	-0.22	-1.02	-1.79	-2.52	-3.21	-3.85	-4.44	-4.96	-5.41	-5.79	-6.07	-6.28	-6.40	-6.46	-6.49	-6.52	-6.54
226	1.32	0.33	-0.64	-1.60	-2.54	-3.46	-4.35	-5.21	-6.03	-6.81	-7.53	-8.20	-8.79	-9.29	-9.69	-9.99	-10.2	-10.3	-10.3	-10.4	-10.4
218	-0.17	-1.26	-2.34	-3.42	-4.48	-5.53	-6.56	-7.56	-8.54	-9.47	-10.4	-11.2	-11.9	-12.6	-13.2	-13.6	-13.9	-14.1	-14.2	-14.2	-14.2
211	-1.18	-2.38	-3.58	-4.77	-5.97	-7.16	-8.33	-9.49	-10.6	-11.7	-12.8	-13.8	-14.8	-15.7	-16.5	-17.1	-17.6	-17.9	-18.0	-18.0	-18.1
203	-1.63	-2.94	-4.25	-5.57	-6.90	-8.23	-9.56	-10.9	-12.2	-13.5	-14.8	-16.0	-17.2	-18.4	-19.4	-20.3	-21.0	-21.5	-21.8	-21.9	-21.9
196	-1.42	-2.83	-4.26	-5.71	-7.17	-8.64	-10.1	-11.6	-13.1	-14.6	-16.1	-17.6	-19.1	-20.5	-21.9	-23.1	-24.2	-25.0	-25.5	-25.7	-25.7
188	-0.44	-1.95	-3.48	-5.04	-6.63	-8.24	-9.88	-11.5	-13.2	-14.9	-16.7	-18.4	-20.2	-21.9	-23.7	-25.3	-26.9	-28.2	-29.1	-29.5	-29.6
180	1.44	-0.16	-1.78	-3.45	-5.15	-6.88	-8.66	-10.5	-12.3	-14.2	-16.2	-18.2	-20.2	-22.3	-24.5	-26.6	-28.7	-30.7	-32.3	-33.3	-33.4
173	4.35	2.69	0.99	-0.76	-2.55	-4.38	-6.27	-8.22	-10.2	-12.3	-14.5	-16.7	-19.0	-21.4	-23.9	-26.5	-29.2	-31.9	-34.6	-36.7	-37.2
165	8.43	6.73	4.98	3.19	1.34	-0.56	-2.53	-4.57	-6.68	-8.88	-11.2	-13.6	-16.1	-18.8	-21.6	-24.6	-27.9	-31.4	-35.2	-39.1	-41.1
158	13.8	12.1	10.4	8.57	6.70	4.77	2.78	0.70	-1.46	-3.72	-6.08	-8.57	-11.2	-14.0	-17.1	-20.3	-24.0	-28.1	-32.8	-38.7	-44.9
150	20.7	19.1	17.3	15.6	13.7	11.8	9.85	7.80	5.66	3.43	1.08	-1.39	-4.02	-6.84	-9.88	-13.2	-16.9	-21.2	-26.2	-33.0	-48.8
143	29.2	27.6	26.0	24.3	22.5	20.7	18.9	16.9	14.9	12.8	10.6	8.22	5.74	3.11	0.27	-2.81	-6.21	-10.1	-14.5	-20.1	-26.2
135	39.5	38.0	36.5	34.9	33.3	31.7	30.0	28.2	26.4	24.5	22.5	20.4	18.2	16.0	13.5	11.0	8.24	5.30	2.16	-0.94	-2.10
128	51.6	50.3	49.0	47.6	46.2	44.7	43.2	41.7	40.1	38.5	36.9	35.2	33.5	31.7	29.9	28.0	26.1	24.3	22.7	21.8	22.0
120	65.6	64.5	63.4	62.2	61.1	59.9	58.7	57.5	56.2	55.0	53.7	52.5	51.2	50.0	48.8	47.7	46.7	46.0	45.5	45.6	46.1
113	81.6	80.7	79.8	78.9	78.0	77.1	76.2	75.3	74.4	73.6	72.7	71.9	71.2	70.5	69.9	69.5	69.2	69.1	69.3	69.7	70.1
105	99.3	98.7	98.1	97.4	96.8	96.2	95.6	95.1	94.5	94.0	93.6	93.2	92.9	92.6	92.5	92.5	92.6	92.9	93.3	93.8	94.2
98	119	118	118	118	117	117	117	116	116	116	116	116	116	116	116	116	117	117	117	118	118
90	140	140	140	139	139	139	139	139	139	139	139	139	139	140	140	140	141	141	141	142	142
83	162	162	162	162	162	162	162	163	163	163	163	163	164	164	164	164	165	165	166	166	166
75	186	186	186	186	186	186	187	187	187	187	187	188	188	188	188	189	189	189	190	190	191
68	210	210	210	210	211	211	211	211	212	212	212	212	213	213	213	213	213	214	214	214	215
60	234	235	235	235	235	236	236	236	236	237	237	237	237	238	238	238	238	238	238	238	239
53	259	260	260	260	261	261	261	261	262	262	262	262	263	263	263	263	263	263	263	263	263
45	285	285	286	286	286	287	287	287	287	288	288	288	288	289	289	289	289	289	289	289	289
38	311	311	312	312	312	313	313	313	314	314	314	314	315	315	315	316	316	316	317	317	317
30	337	338	338	338	339	339	340	340	340	341	341	341	342	342	343	343	343	344	344	345	345
23	364	365	365	365	366	366	367	367	368	368	368	369	369	370	370	371	371	372	372	373	373
15	392	392	392	393	393	394	394	395	395	396	396	397	397	398	398	399	399	400	400	401	401
8	419	420	420	421	421	422	422	423	423	424	424	425	425	426	426	427	427	428	428	429	429
0	447	448	448	449	449	450	450	451	451	452	452	453	453	454	454	455	455	455	456	456	457

Black-Scholes table. The **blue glow on the right edge** represents the user's Assumed Probability Density: brighter glow indicates higher probability of the stock ending at that price.

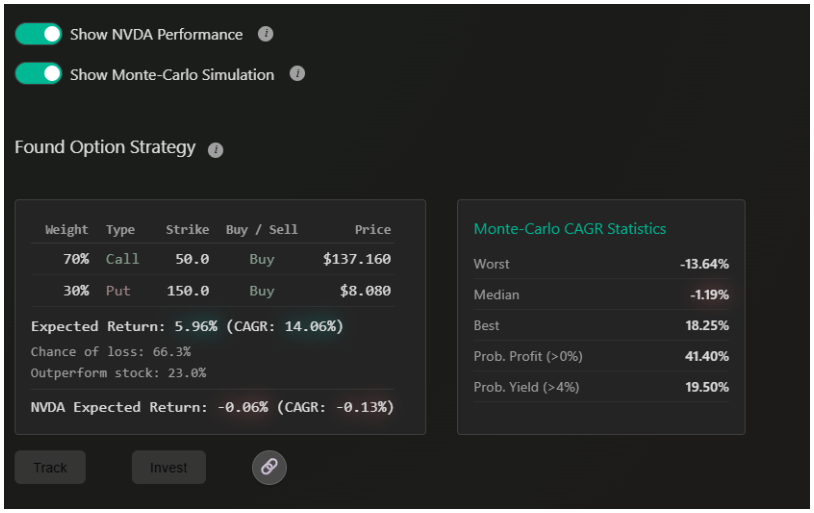
- **Found Option Strategy:** The optimal spread with its legs, percentage weights, and expected return.
- **Risk Profile Comparison:** Overlay of the user's risk tolerance $\hat{r}_{tol}$ (red) and the strategy's actual risk profile $\hat{r}_c$ (blue).



The blue line (strategy) strictly dominates the red line (tolerance).

- **Underlying Performance Toggle:** "Show {ticker} Performance" reveals the stock's expected return and risk profile for comparison.
- **Outperform Probability:** The probability that the found strategy outperforms simply holding the stock, computed via spline intersections with the PDF.

- **Chance of Loss:** The probability that the strategy results in a net loss.
- **Monte-Carlo CAGR Statistics:** Simulated compound annual growth rate—see Risk Theory (Part IV) for why this matters beyond expected return.



Found strategy card with Monte-Carlo CAGR statistics.

8 Max Min Mode

Max Min takes a fundamentally different approach to portfolio construction. Instead of maximizing expected return (which depends on the accuracy of a user-supplied probability distribution), it maximizes the **worst-case outcome** (minimax optimization):

$$\max_{\mathbf{w}} \min_{S_T \in [0,\infty)} \sum_i w_i \, g_i(S_T)$$

where $\mathbf{w}$ is the weight vector over available option legs.

This is a **distribution-free** criterion: it makes no assumption about where the stock will end up. It answers the question: "If I had to prepare for the absolute worst, what is the strategy whose floor is the highest?"

8.1 Single Stock Mode

8.1.1 Workflow

1. **Enter a Ticker:** Same data pipeline as the other modes (price, options chain, implied PDFs).
2. **Select an Expiration Date:** Choose from the available expirations list.
3. **Launch Search:** The optimizer evaluates all spread combinations and finds the one whose minimum across all possible stock prices S_T is maximized.

8.1.2 Mathematical Formulation

For a single stock with available option payoff functions $\{g_i\}_{i\in[n]}$, the optimization problem is:

$$\max_{s\in\mathcal{S}} \min_{x\in\mathbb{R}} g_s(x) = \max_{\substack{o_i\in\mathcal{O} \\ w_i\in\mathbb{R} \\ \sum w_i=1}} \min_{x\in\mathbb{R}} \sum_i w_i \cdot g_{o_i}(x) \tag{11}$$

Because each g_i is piecewise linear (for vanilla options), the minimum over S_T occurs at a vertex of the piecewise-linear surface. The problem reduces to a finite-dimensional linear program.

8.2 Multi Stock Mode

Multi mode extends the minimax framework across multiple tickers. The user builds a portfolio by adding stocks one at a time, each with its own expiration and option chain.

8.2.1 Workflow

1. **Add Stocks:** Enter a ticker and click "Add." Repeat for each stock.
2. **Select Expirations:** Choose an expiration date per stock.
3. **Launch Search:** The optimizer finds both the cross-asset allocation *and* the per-stock option spread that maximizes the portfolio's worst case.

8.2.2 Cross-Asset Diversification

With M stocks, each having payoff functions $\{g_{m,i}\}$, the problem becomes:

$$\max_{\mathbf{w}_1,\ldots,\mathbf{w}_M,\,\boldsymbol{\alpha}\in\Delta^M} \; \min_{S_{T,1},\ldots,S_{T,M}} \; \sum_{m=1}^{M} \alpha_m \sum_i w_{m,i}\, g_{m,i}(S_{T,m})$$

where $\boldsymbol{\alpha}$ is the inter-stock capital allocation and $\mathbf{w}_m$ is the intra-stock option spread. Because the stocks are independent, the global minimum decomposes into per-stock minima, and the outer allocation $\boldsymbol{\alpha}$ optimally diversifies across the per-stock floors.

Feature Isolation: Actions in Single mode (selecting tickers, dates) never affect the Multi mode state, and vice versa. This is enforced at the UI level.

Note: The Max Min optimization algorithm is currently in development. The UI for both sub-modes is complete and functional.

8.3 Soft Minimum Optimization (Percentile-Based Targets)

While pure Max Min optimizes the absolute worst-case outcome (the 0th percentile), users may prefer to relax this extreme conservatism and instead optimize a higher percentile—such as the 5th or 10th percentile. This approach is known as **Soft Minimum maximization**, **Max Min Soft** or **Value-at-Risk (VaR) optimization**.

8.3.1 When to Use Soft Minimum

- **Acceptable Risk Tolerance:** You are willing to accept a 5% chance of outcomes below a threshold to improve average performance for the other 95% of scenarios. - **Realistic Worst-Case Budgets:** The absolute worst-case floor may be so expensive (or impossible) to guarantee that you prefer to focus on a more attainable probabilistic floor. - **Long-Term Investing:** For multi-period strategies, the 5th percentile often provides more meaningful protection than the theoretical minimum (which might represent a scenario with 0.001% probability).

8.3.2 How It Works

Instead of optimizing $\max(\min g)$, soft minimum optimization solves:

$$\max_{\mathbf{w}} \left(\text{Value-at-Risk}_{p^*}[g]\right)$$

where p^* is the chosen percentile level (e.g., $p^* = 0.05$ for the 5th percentile). This means: "Maximize the return level such that 5% of scenarios breach below it."

Under the hood:

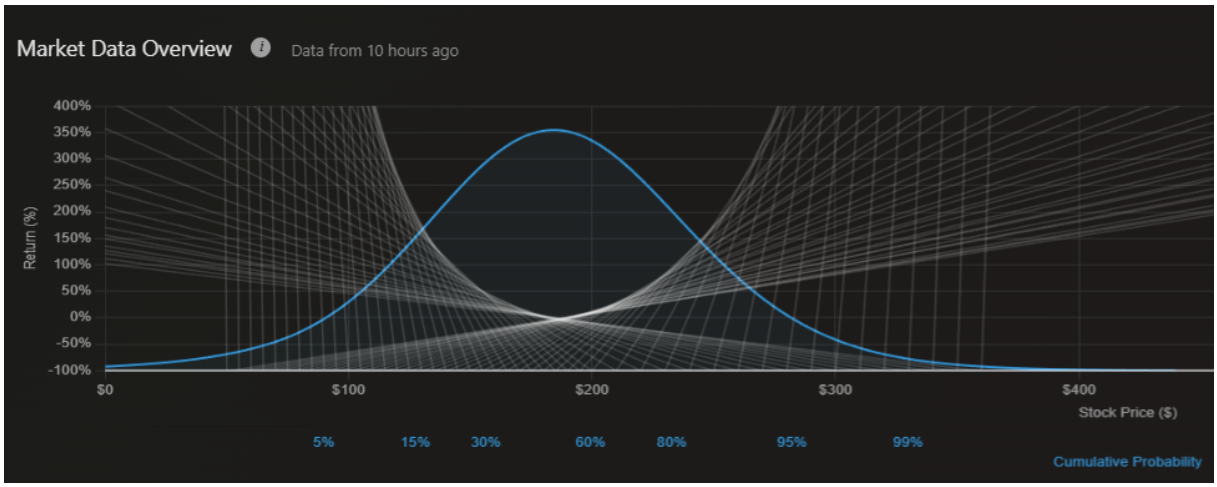
1. The optimizer identifies the worst $p^* \times 100\%$ of stock price outcomes (the "sacrifice zone").
2. Within this zone, the strategy can yield any payoff (potentially zero or negative, even bankrupcy if not protected against it).
3. For the remaining $100(1 - p^*)\%$ of scenarios, the optimizer maximizes the floor.
4. The freed-up budget from sacrificing the worst outcomes can significantly improve protection for the preserved percentiles.

8.3.3 Hybrid Approach: VaR Floor with Growth

Users can also combine a soft-minimum floor with upside potential. Instead of capping the protected percentiles at a flat floor, the optimizer can use the Farmer approach (from Probability & Risk mode) to create a curved payoff that still respects the VaR floor. This results in: a guaranteed 5th percentile return, with probability-weighted growth opportunities above that floor.

9 Market Data Overview

When a ticker and expiration are selected, the Market Data Overview chart is displayed:



Market Data Overview: thin white lines show individual option performance curves; the colored curve is the Implied PDF (peak $\sim$ \$180); the CDF is overlaid.

10 Implied Probability Distributions

10.1 What the Market "Thinks"

The Implied PDF is derived from current option prices using the following pipeline:

1. Compute implied volatility (IV) for each out-of-the-money option using the Black-Scholes formula.
2. Fit a **Quadratic Curve** to IV as a function of strike: $IV(K) = aK^2 + bK + c$.
3. Use the smoothed IV to generate dense Black-Scholes prices $C(K)$ across a fine strike grid.

4. Derive the PDF via the **Breeden-Litzenberger** finite-difference formula:

$$\mu(K) \approx e^{rT} \frac{C(K + \Delta K) - 2C(K) + C(K - \Delta K)}{(\Delta K)^2} \tag{12}$$

This **IV Smoothing** method avoids the instabilities (Runge's phenomenon) of fitting splines directly to noisy option prices.

10.2 Dynamic Implied Volatility

The standard Black-Scholes model uses a *constant* IV per option leg, calibrated from current market prices. This "static IV" assumption means the model predicts the same volatility regardless of where the stock moves. In reality, implied volatility varies with the stock price — a phenomenon known as the **volatility smile** or **volatility skew**.

CallCulator's **Dynamic IV** feature models this by adjusting each leg's IV based on the stock's displacement from its current price:

$$\sigma^{\text{dyn}}(S) = \sigma_{\text{base}} \times \left(1 + \alpha \cdot \Delta + \beta \cdot \Delta^2\right) \tag{13}$$

where $\Delta = (S - S_0)/S_0$ is the relative price change, α is the **skew** parameter, and β is the **curvature** parameter.

Why does IV change with price? Three effects drive the relationship between stock price and IV:

1. **Crash fear and put demand (Skew):** When stocks fall, investors rush to buy protective puts, driving up option prices and thus IV. When stocks rise, this panic subsides and IV compresses. This directional asymmetry is captured by the skew parameter ($\alpha < 0$ for equities).

2. **Leverage effect (Skew):** A falling stock price increases the company's debt-to-equity ratio, making it objectively riskier. Higher risk translates to higher expected future volatility.

3. **Regime shifts (Curvature):** Any large move — up *or* down — signals that the market's volatility regime may have changed. A stock that jumps 30% in either direction is clearly in a more volatile state than one that barely moves. This symmetric effect is captured by the curvature parameter ($\beta > 0$).

Practical examples:

- With $\alpha = -1, \beta = 0$: A 10% stock drop increases IV by 10% of its base value (e.g., 30% $\rightarrow$ 33%). A 10% stock rise *decreases* IV by 10% (30% $\rightarrow$ 27%).

- With $\alpha = -1, \beta = 0.5$: A 10% drop increases IV by 10.5% (skew + curvature). A 10% rise decreases IV by only 9.5% (skew partially offset by curvature).

- With $\alpha = 0, \beta = 1$: Pure smile — IV increases symmetrically for any large move.

Initial condition guarantee: At the current stock price ($S = S_0$), $\Delta = 0$, so $\sigma^{\text{dyn}} = \sigma_{\text{base}}$. This means the Black-Scholes table always shows exactly 0% performance at today's price and time, regardless of the skew and curvature settings.

Usage in CallCulator: Enable the **Dynamic IV** toggle in Search Parameters and set the **Skew** value (typical equity values: -2 to -0.5). The Black-Scholes table will then reflect how your strategy performs under a more realistic volatility surface. Use the **Show IV X-ray** button (hold to reveal) above the table to see the effective IV at each cell.

11 Data Pipeline

- **Data Manager:** Serves market data to the frontend. Checks S3 data freshness and triggers updates if stale.
- **Data Updater:** Fetches live data from the Alpaca API when triggered.
- **Caching:** Three-layer architecture: S3 (backend), CloudFront (edge), Browser (30-second window).
- **Implied PDFs stay in sync:** Any options refresh automatically triggers a recalculation of all implied PDFs for that ticker.
- **Thundering Herd Protection:** Uses `isBeingUpdated` flag + Probabilistic Early Expiration (PEE) to prevent cache stampedes.

Part IV
Risk Theory

12 Risk Profiles

Every investment carries risk. CallCulator quantifies risk through two mathematical objects:

12.1 Personal Risk Tolerance Profile

A function $\hat{r}_{tol} : [0, 1] \to \mathbb{R}$ mapping cumulative probability λ to the minimum acceptable return at that probability level.

Example: "I never want to lose more than 25% (at $\lambda = 0$), in 50% of cases I want to profit (at $\lambda = 0.5$, return ≥ 0), and my best case should be at least +100% (at $\lambda = 1$)."

12.2 Investment Risk Profile

For a strategy with performance function $p_c(x)$ and assumed PDF $\mu(x)$, the risk profile is:

$$\hat{r}_c(\lambda) = p_c\big(\Psi^{-1}(\lambda)\big), \qquad \Psi(x) := \int_0^x \mu(y)\,\mathrm{d}y \tag{14}$$

where Ψ is the cumulative distribution function. This maps each cumulative probability level $\lambda \in [0, 1]$ to the corresponding return of the strategy.

12.3 Matching Profiles

A strategy is **valid** if its risk profile dominates the tolerance at every point:

$$\hat{r}_c(\lambda) \geq \hat{r}_{tol}(\lambda) \quad \forall\, \lambda \in [0, 1]$$

The optimization engine searches for the valid strategy with the highest expected return.

13 On the Merit of Spreads

13.1 The Single-Step Argument Against Spreads

For a single, isolated investment, the expected return of a spread is:

$$\mathbb{E}\Big[\sum_i w_i\, g_i\Big] = \sum_i w_i\, \mathbb{E}[g_i] \tag{15}$$

Since this is a convex combination of real numbers, its maximum occurs at the boundary ($w_j = 1$ for some j). Therefore, in a single-step game, a single option always has the highest expected return. Spreads appear pointless.

13.2 The Multi-Step Counter-Argument: Geometric Growth

Investing is not a single event—it is a **repeated game**. In repeated games, maximizing the arithmetic expectation can lead to guaranteed ruin.

13.2.1 Alice and Bob: A Story of Ruin

Alice offers Bob a coin (80% Heads, 20% Tails):

- Heads: +100% (investment doubles)
- Tails: −100% (investment lost)

Rule: Bob must reinvest his entire bankroll.

Expected return per flip: +60%. But if Bob plays long enough:

$$P(\text{at least one Tails in } n \text{ flips}) = 1 - 0.8^n \xrightarrow{n\to\infty} 1$$

Bob is bankrupt with probability 1. The arithmetic expectation (+60%) is misleading; the **geometric growth rate** is $\ln(0) = -\infty$ on a Tails outcome.

13.2.2 Position Sizing Saves the Day

By betting only a fraction f of his capital, Bob's geometric growth rate per flip becomes:

$$G(f) = 0.8\, \ln(1+f) + 0.2\, \ln(1-f)$$

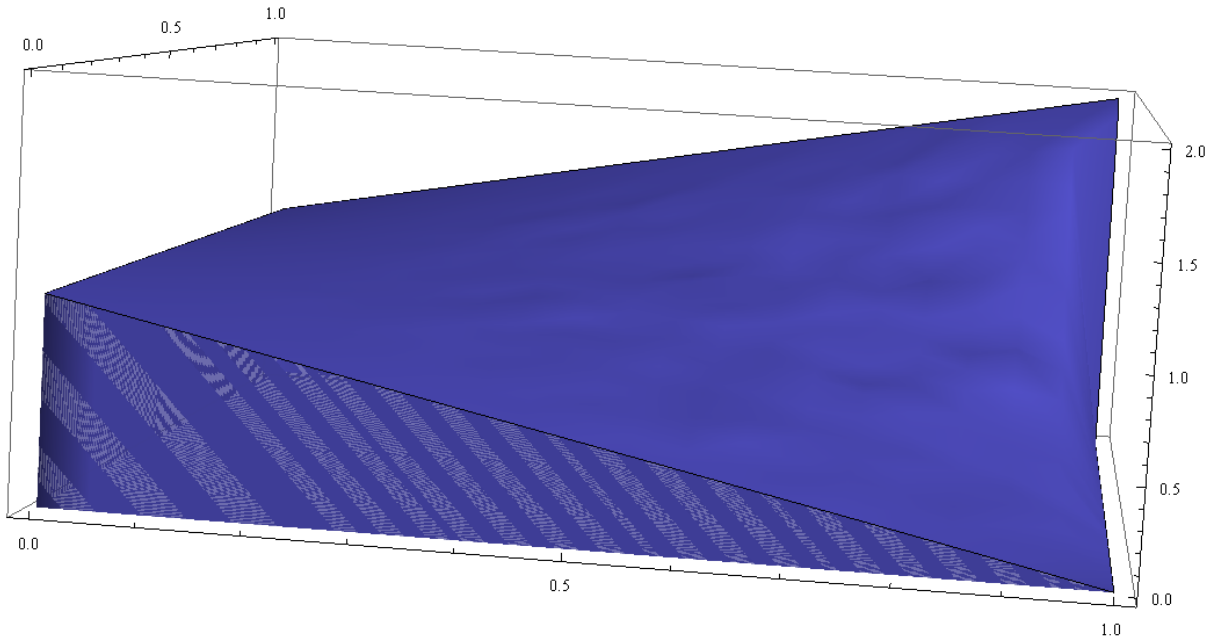
Maximizing $G(f)$ yields the **Kelly fraction**—the optimal bet size.

13.3 Beyond the Coin Flip: The Ergodicity Problem

This coin-flip example might seem extreme, but the core issue persists in almost all repeated investing. The mathematical term for this is **ergodicity breaking**.

In financial markets, the **Expected Return** (average of all possible outcomes) is often very different from the **Median Outcome** (what actually happens to most people). A strategy might have a positive Expected Return but still lead to a loss for 90% of investors over time because a few massive winners skew the average.

When you invest repeatedly (re-investing your gains), you don't experience the average of parallel universes; you experience one single path through time. If that path hits zero, you are out of the game. This is why maximizing Expected Return is not enough—you must manage the risk of the "time-average" diverging from the "ensemble-average."



Simulation of average return per flip vs. fraction of capital wagered.

13.4 Monte-Carlo Simulation: Quantifying Hidden Risk

Expected return alone is an insufficient metric for investment decisions in a repeated-game context. Two strategies with identical $\mathbb{E}[g]$ can have vastly different geometric growth rates depending on the variance and tail behavior of their return distributions.

A **Monte-Carlo CAGR simulation** would expose this by random sampling:

1. Draw N samples $\{x_j\}_{j=1}^{N}$ from the user's PDF $\mu(x)$.
2. For each sample, compute the strategy return $g(x_j)$.
3. Simulate T rounds of compounding: $W_T = W_0 \prod_{t=1}^{T}(1 + g(x_{\sigma(t)}))$ where σ is a random draw.
4. Report the annualized geometric mean: $\text{CAGR} = (W_T/W_0)^{1/T} - 1$, averaged over many runs.

However, Monte-Carlo is stochastic: each run gives slightly different results, tail events are under-sampled, and convergence is slow ($\mathcal{O}(1/\sqrt{K})$ for K paths). CallCulator replaces this with a **deterministic** alternative.

13.5 FFT Convolution: Deterministic CAGR Distributions

Instead of sampling random paths, CallCulator computes the **exact** compound return distribution by exploiting the Convolution Theorem on log-return probability densities.

13.5.1 Core Insight: Quantile Wealth Multipliers

The strategy's risk profile is a **return quantile function** $Q : [0, 1] \to \mathbb{R}$, mapping each percentile p to the return at that probability level. From this, we define the **wealth multiplier quantile function**:

$$W(p) = 1 + Q(p), \qquad p \in [0, 1] \tag{16}$$

Each percentile gives a *different* wealth multiplier: $W(0)$ is the worst-case multiplier (e.g. 0.6 if the worst loss is -40%), while $W(1)$ is the best-case (e.g. 3.0 for $+200\%$).

When we compound over N independent periods, the total wealth is the product of the single-period multipliers. Taking the logarithm converts this product into a sum:

$$\mathcal{W}_N = \prod_{t=1}^{N} W_t \qquad \Longrightarrow \qquad \ln \mathcal{W}_N = \ln\Big(\prod_{t=1}^{N} W_t\Big) = \sum_{t=1}^{N} \ln W_t \tag{17}$$

This transforms the problem of multiplying random variables into the simpler problem of adding them.

The probability distribution of a sum of independent random variables is the **convolution** of their individual distributions. (Intuition: to get a sum $Z = X + Y$, for every value x that X can take, Y must take exactly $Z - x$; summing these mutually exclusive possibilities weighted by their probabilities is the definition of convolution).

In the frequency domain (Fourier space), this complex convolution operation becomes simple pointwise multiplication:

$$\widehat{p_L^{*N}} = \big(\hat{p}_L\big)^N \tag{18}$$

where p_L is the single-period log-return density. To compute this efficiently, we approximate the continuous density with a discrete histogram and use the Fast Fourier Transform (FFT).

13.5.2 Protocol

Given the strategy's return quantile function Q, the number of compounding periods N, and a reinvestment fraction f:

1. **Discretize** the quantile function at M bin-center percentiles to obtain discrete quantile returns: $r_k = Q\big(\frac{k+0.5}{M}\big)$.
2. **Form quantile wealth multipliers**: $W_k = 1 + r_k/100$. Each W_k corresponds to a different percentile of the outcome distribution.
3. **Apply partial reinvestment**: $W_k^{(f)} = (1 - f) + f \cdot W_k$ and clip $W_k^{(f)} > 0$. Only a fraction f of capital is exposed each period.
4. **Log-transform** to discrete quantile log-returns: $\ell_k = \ln W_k^{(f)}$.
5. **Build a normalized histogram PDF** of the log-returns over power-of-2 bins. Since the quantile samples are uniformly spaced in probability, each contributes equal mass to the histogram.
6. **Zero-pad** to length $\geq N \cdot N_b$ to prevent circular wraparound.
7. **Forward FFT** (Cooley-Tukey radix-2), raise spectrum to power N, **inverse FFT**.
8. **Map convolved bins to CAGR**: $\mathrm{CAGR}_m = \exp(S_m/N) - 1$ where S_m is the total log-return at bin m.
9. **Build CDF** to produce the final CAGR quantile function—a mapping from percentile to compound growth rate, describing N-period compounded outcomes.

13.5.3 Advantages over Monte-Carlo

- **Deterministic**: Identical inputs always produce identical outputs.
- **Exact tails**: Unlike Monte-Carlo which under-samples rare events, the FFT method resolves the full distribution uniformly—critical for quantifying ruin risk.
- **Speed**: $\mathcal{O}(M \log M)$ regardless of N, compared to $\mathcal{O}(K \cdot N)$ for K Monte-Carlo paths.
- **Multi-N animation**: CallCulator computes profiles for $N = 1, 2, \ldots, 20$, visualizing how the CAGR quantile function concentrates over time.

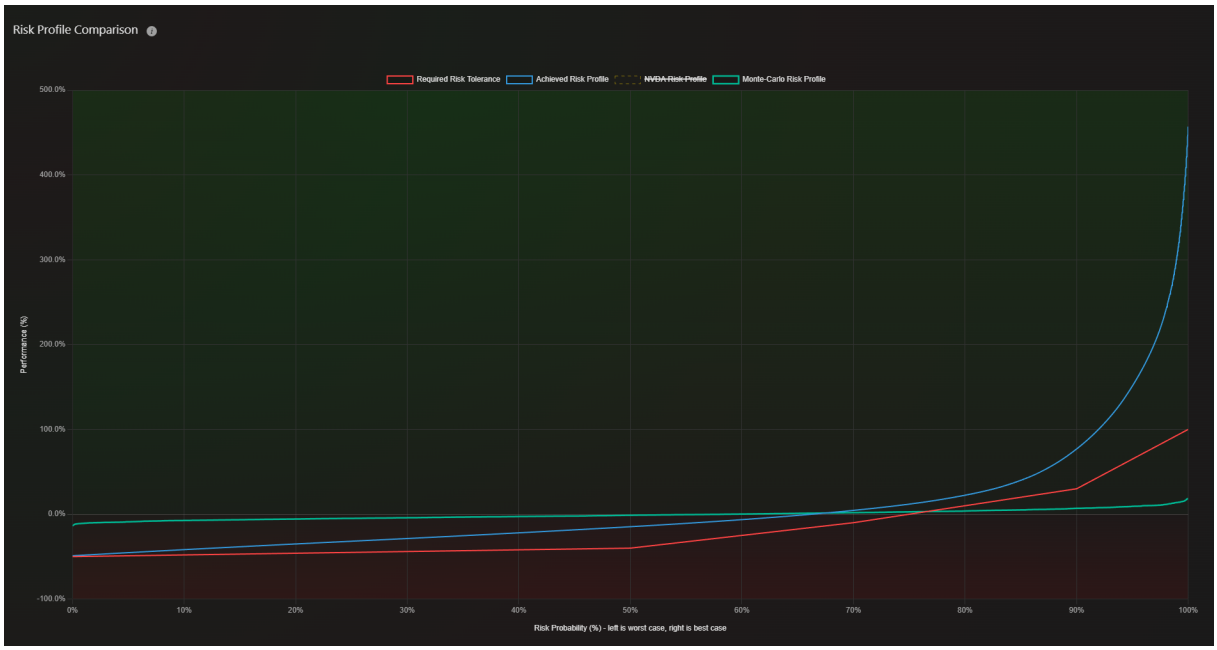
13.6 Kelly Criterion: Optimal Reinvestment Fraction

The FFT convolution requires a reinvestment fraction f. The **Kelly Criterion** finds the growth-optimal fraction f^* by maximizing the expected log-growth rate over the quantile returns:

$$f^* = \text{argmax}_{f \in [0, f_{\max}]} \; G(f) = \text{argmax}_f \; \frac{1}{M} \sum_{k=0}^{M-1} \ln\bigl(1 + f \cdot r_k\bigr) \tag{19}$$

where r_k are the single-period fractional returns sampled from the return quantile function. The upper bound $f_{\max} < -1/r_{\min}$ ensures solvency ($1 + f \cdot r_{\min} > 0$). Since $G(f)$ is concave, f^* is unique and is found via golden-section search.

CallCulator feeds f^* into the FFT convolution as the reinvestment parameter. The Kelly fraction is also displayed as an informational metric alongside the CAGR statistics, giving users insight into how much of their capital should be allocated to the strategy for optimal long-term growth.



Risk Profile Comparison with FFT-computed CAGR profiles.

13.7 Log-Optimal Strategy: Maximizing Median CAGR

The Probability & Risk optimization engine (Section 3.2) maximizes the **expected return** $\mathbb{E}_P[g]$—a linear objective. As the Alice & Bob story and the ergodicity discussion demonstrate, expected return can be a misleading metric for repeated investment. A strategy with the highest expected return may have a median compound growth rate of zero.

An alternative objective is to maximize the **Median Monte-Carlo CAGR** directly. Mathematically, this is equivalent to maximizing the expected logarithm of wealth:

$$\max_g \int_0^\infty \ln(g(x))\, P(x)\, dx \tag{20}$$

subject to the budget constraint $\int_0^\infty g(x)\, Q(x)\, dx = K$.

13.7.1 The Log-Optimal Payoff

Solving via the Lagrangian method yields the optimal payoff:

$$g^*(x) = \frac{1}{\lambda} \cdot \frac{P(x)}{Q(x)} \tag{21}$$

where λ is the Lagrange multiplier (budget normalizer) and $P(x)/Q(x)$ is the likelihood ratio $\Lambda(x)$.

This is fundamentally different from the expected-return-maximizing "Sniper" strategy:

- **Sniper (Max Expected Return):** Concentrates all capital at a single spike at argmax $\Lambda(x)$. Highest average return, but median wealth of \$0 over time (most spikes miss).
- **Farmer (Max Median CAGR):** Spreads capital across the *entire* state space proportional to $\Lambda(x)$. Lower average return, but highest median wealth over time. The payoff never reaches zero, avoiding the ruin that destroys geometric growth.

13.7.2 Adding the Risk Tolerance Floor

When a risk tolerance $T(q)$ is imposed (minimum required payoff at quantile q), we must translate this requirement into the price domain via $\tilde{T}(x) := T(F_\Lambda(\Lambda(x)))$. The constrained log-optimal solution becomes:

$$g^*(x) = \max\left(\tilde{T}(x),\ \frac{1}{\lambda} \cdot \frac{P(x)}{Q(x)}\right) \tag{22}$$

This is a **threshold strategy**: below the threshold, you buy "insurance" (paying the minimum cost to satisfy $\tilde{T}$); above the threshold, you allocate according to the Kelly-optimal Λ-shaped profile. In financial terms, this resembles a CPPI (Constant Proportion Portfolio Insurance) strategy—a protective floor plus a growth allocation.

13.7.3 Computing λ: Newton-Raphson on the Budget Equation

The multiplier λ is found by solving the budget equation $f(\lambda) = K$, where:

$$f(\lambda) = \int_0^\infty \max\left(\tilde{T}(x),\ \frac{1}{\lambda} \cdot \frac{P(x)}{Q(x)}\right) Q(x)\, dx \tag{23}$$

The derivative is:

$$f'(\lambda) = -\frac{1}{\lambda^2} \int_{\Omega_\lambda} P(x)\, dx \tag{24}$$

where Ω_λ is the "active set" of states where the Kelly bet exceeds the floor. Since f is monotonically decreasing and convex, Newton-Raphson converges rapidly (typically 3–4 iterations). CallCulator's polynomial spline infrastructure enables exact evaluation of both $f(\lambda)$ and $f'(\lambda)$ via coefficient-shifting integration, avoiding discretization noise entirely.

13.8 Implications for Options

A spread is precisely this: instead of betting everything on a single call option (maximizing single-step expected return), a spread allocates capital across multiple payoff profiles, avoiding absorbing barriers (total loss) and maximizing long-term geometric growth. This is why CallCulator's risk-constrained optimization produces spreads—they are the geometrically optimal strategy for repeated investing.

14 Mathematical Constructions: The Six Optimization Regimes

The various optimization objectives available in CallCulator (and planned for future implementation) can be understood as distinct mathematical "regimes," each with a characteristic optimal payoff structure. These regimes form a hierarchy from micro-arbitrage to macro-strategic allocation.

14.1 Regime 0: The Scavenger (Micro-Arbitrage)

Before constructing a global strategy based on your belief $P(x)$, the Scavenger exploits static mispricings within the market data itself—deviations between observed option prices and the fitted theoretical curve $Q(x)$.

14.1.1 Residual Mispricing

Let $\{(K_i, p^i_{mkt})\}$ be the observed market prices and $C_{fit}(K)$ be the theoretical price curve from the IV Smoothing pipeline. The **residual** for each option is:

$$\epsilon_i = p^i_{mkt} - C_{fit}(K_i) \tag{25}$$

Strategy:

- **Buy** options with $\epsilon_i < -\theta$ ("cheap")
- **Sell** options with $\epsilon_i > \theta$ ("expensive")

where θ is a transaction cost threshold.

14.1.2 Butterfly Spreads and Neighbor Selection

To isolate a mispricing without taking directional risk, construct a butterfly spread. If option K_i is cheap, buy it and sell neighbors K_{i-m} and K_{i+n} to hedge.

Neighbor Selection Rule: Expand the window $[K_{i-m}, K_{i+n}]$ outward until you reach strikes where:

$$\text{sgn}(\epsilon_{i\pm k}) \neq \text{sgn}(\epsilon_i) \quad \text{or} \quad |\epsilon_{i\pm k}| < \delta \tag{26}$$

This ensures you hedge a "cheap" option against "fair" or "expensive" ones, maximizing residual capture.

14.1.3 Boosting Macro Strategies

The Scavenger's role is not standalone—it **boosts** the macro regimes (1–5). When discretizing a theoretical payoff $g^*(x)$ into actual options, preferentially choose "cheap" outliers. This lowers the actual cost below the theoretical cost, freeing up budget to scale up the position or improve the floor.

14.2 Regime 1: The Sniper (Unconstrained Maximization)

Objective: Maximize expected return with a fixed budget K.

$$\max_g \int_0^\infty g(x)P(x)\,dx \quad \text{s.t.} \quad \int_0^\infty g(x)Q(x)\,dx = K \tag{27}$$

Optimal Payoff: A Dirac delta (infinitely sharp spike) at the point maximizing the likelihood ratio:

$$x^* = \text{argmax}_x \frac{P(x)}{Q(x)} \tag{28}$$

Interpretation: The Sniper bets everything on the single most mispriced outcome. In practice, this is approximated by a very tight butterfly spread centered at x^*. This strategy has the highest possible expected return but a median outcome of \$0 over repeated trials (most spikes miss).

14.3 Regime 2: The Insurer + Sniper (Constrained Optimization)

Objective: Maximize expected return subject to a risk tolerance floor $T(q)$.

$$\max_g \mathbb{E}_P[g] \quad \text{s.t.} \quad \forall q \in [0,1] : R_g^{-1}(q) \geq T(q) \tag{29}$$

Optimal Payoff: A two-part portfolio:

$$g^*(x) = g_{safe}(x) + g_{spec}(x) \tag{30}$$

where:

- **Safety Portfolio:** The cheapest portfolio satisfying $T(q)$, constructed via the Hardy-Littlewood rearrangement:

$$g_{safe}(x) = \tilde{T}(x) := T(F_\Lambda(\Lambda(x))) \tag{31}$$

  Why does g_{safe} depend on Λ and Q? The risk tolerance $T(q)$ dictates the distribution of payouts under the investor's belief P (e.g., "the worst 5% of outcomes must pay at least \$90"), but does not dictate *which* share prices deliver them. To maximize the remaining budget for the speculative "Sniper" bet, we must satisfy $T(q)$ as cheaply as possible. We do this by assigning the highest required outcomes (top of T) to the most cost-efficient states (highest $\Lambda = P/Q$), and the lowest required outcomes to the most expensive states (lowest Λ).

- **Speculative Portfolio:** Any remaining budget is allocated to the Sniper bet (Dirac delta at $\operatorname{argmax} \Lambda$).

Interpretation: "Insure then speculate." First, guarantee the risk floor at minimum cost. Then, use the surplus for maximum expected return.

14.4 Regime 3: The Farmer (Log-Optimal / Median CAGR Maximization)

Objective: Maximize the expected geometric growth rate (equivalently, the Median CAGR).

$$\max_g \int_0^\infty \ln(g(x))\, P(x)\, dx \quad \text{s.t.} \quad \int_0^\infty g(x) Q(x)\, dx = K \tag{32}$$

Optimal Payoff: Proportional to the likelihood ratio:

$$g^*(x) = \frac{1}{\lambda} \cdot \frac{P(x)}{Q(x)} = \frac{1}{\lambda}\Lambda(x) \tag{33}$$

Interpretation: Unlike the Sniper (which concentrates at a single spike), the Farmer spreads capital across the *entire* state space in proportion to the mispricing signal $\Lambda(x)$. The payoff never reaches zero, avoiding the ruin that destroys geometric growth. This strategy has a lower expected return than the Sniper but the highest median wealth over repeated trials.

14.4.1 Risk-Tolerance Constrained Farmer

Adding a risk tolerance floor $\tilde{T}(x) := T(F_\Lambda(\Lambda(x)))$ translating quantile $T(q)$ to price domain:

$$g^*(x) = \max\left(\tilde{T}(x),\ \frac{1}{\lambda}\Lambda(x)\right) \tag{34}$$

This is a **threshold strategy**: below the threshold, you buy insurance (paying the minimum cost to satisfy $\tilde{T}$); above the threshold, you allocate according to the Kelly-optimal Λ-shaped profile. This resembles a CPPI (Constant Proportion Portfolio Insurance) strategy.

14.4.2 Solving for λ: Newton-Raphson

The Lagrange multiplier λ is found by solving the budget equation:

$$f(\lambda) = \int_0^\infty \max\left(\tilde{T}(x),\ \frac{1}{\lambda}\Lambda(x)\right) Q(x)\,dx = K \tag{35}$$

The derivative is:

$$f'(\lambda) = -\frac{1}{\lambda^2}\int_{\Omega_\lambda} P(x)\,dx \tag{36}$$

where $\Omega_\lambda = \{x \mid \frac{1}{\lambda}\Lambda(x) > \tilde{T}(x)\}$ is the "active set." Since f is monotonically decreasing and convex, Newton-Raphson converges rapidly (typically 3–4 iterations).

CallCulator's polynomial spline infrastructure enables exact evaluation of both $f(\lambda)$ and $f'(\lambda)$ via coefficient-shifting integration, avoiding discretization noise entirely.

14.5 Regime 4: The Bunker (Max Min Optimization)

Objective: Maximize the worst-case outcome.

$$\max_g \left(\min_x g(x)\right) \quad \text{s.t.} \quad \text{Cost}[g] = K \tag{37}$$

Optimal Payoff: A constant (flat line):

$$g^*(x) = M \quad \text{(constant)} \tag{38}$$

Proof: Write $g(x) = M + \epsilon(x)$ where $\epsilon(x) \geq 0$ is any upside. The cost is:

$$\text{Cost} = M\int Q(x)\,dx + \int \epsilon(x)Q(x)\,dx = K \tag{39}$$

To maximize M, minimize the "Cost of Upside." Since $\epsilon \geq 0$ and $Q > 0$, the integral is minimized only when $\epsilon(x) = 0$ everywhere.

Implementation: A Box Spread (Long Call K_1, Short Call K_2, Long Put K_2, Short Put K_1), which creates a synthetic risk-free bond with constant payoff $K_2 - K_1$.

14.6 Regime 5: The Sentinel (Soft Minimum / Percentile-Based Targets)

The pure Bunker is extremely conservative—it protects against *every* possible outcome, including those with near-zero probability. The **Sentinel** relaxes this by maximizing a higher percentile (e.g., 5th or 10th) rather than the absolute minimum.

Objective: Maximize the Value-at-Risk at level p^*:

$$\max_g \left(q_g^*\right) \quad \text{where} \quad q_g^* = \inf\left\{z : P(g(x) \leq z) \geq p^*\right\} \tag{40}$$

Interpretation: The Sentinel guards the important percentiles (e.g., 95% of outcomes) while accepting a "sacrifice zone" (the worst 5%). This frees up budget to improve the protected floor.

Hybrid Approach: Combine a soft-minimum floor with the Farmer's growth allocation:

$$g^*(x) = \max\left(M, \frac{1}{\lambda}\Lambda(x)\right) \tag{41}$$

where M is the soft-minimum floor (determined by the p^* percentile constraint) and the remaining budget is spent on the $\frac{1}{\lambda}\Lambda(x)$ curve above it. This creates a guaranteed floor for the protected percentiles with probability-weighted growth on top.

14.7 Summary: Comparison of All Regimes

Regime	Optimization	Objective	Optimal Payoff $g^*(x)$
0: Scavenger	max residual $\lvert\epsilon_i\rvert$	$\max \lvert\epsilon_i\rvert$	Outlier isolation
1: Sniper	$\max_g$	$\mathbb{E}_P[g]$	Dirac-delta at argmax($\Lambda(x)$)
2: Insurer + Sniper	$\max_{\substack{s\in\mathcal{S} \\ \forall q\in[0,1]: R_s^{-1}(q)\geq T(q)}}$	$\mathbb{E}_P[g]$	Floor T + Dirac-delta at argmax($\Lambda(x)$)
3: Farmer	$\max_g$	$\mathbb{E}_P[\ln(g)]$	$\frac{1}{\lambda}\Lambda(x)$
4: Bunker	$\max_g$	$\min_x g(x)$	Constant (Box Spread)
5: Sentinel	$\max_g$	$\text{VaR}_{p^*}[g]$	Sacrifice zone floor M + growth

Current Implementation: CallCulator's Probability & Risk mode implements Regime 2 (Insurer + Sniper). The "Use Monte-Carlo Median as objective function" toggle is a placeholder for Regime 3 (Farmer). Max Min mode implements Regime 4 (Bunker), with Regime 5 (Sentinel) planned for future implementation.

15 From Continuous Payoff to Discrete Portfolio

The preceding section outlines the analytically optimal payoff function $g^*(x)$ for each regime as a *continuous* function of the underlying price x. In practice, however, one cannot purchase a continuous payoff. The available instruments are a finite set of call and put options at discrete strikes $\{K_1, K_2, \ldots, K_N\}$, each with a market price p_i^{mkt}. This section formalizes the **discretization problem**: given $g^*(x)$, find the portfolio of available options that best approximates it.

15.1 General Framework

Let $\mathcal{H} = \{h_1(x), h_2(x), \ldots, h_N(x)\}$ be the set of available option payoff functions, where each $h_i(x)$ is either a call payoff $\max(x - K_i, 0)$ or a put payoff $\max(K_i - x, 0)$. A portfolio is defined by a weight vector $\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$, where $w_i > 0$ denotes a long position and $w_i < 0$ a short position. The portfolio payoff is:

$$\hat{g}(x) = \sum_{i=1}^{N} w_i \, h_i(x) \tag{42}$$

Since each $h_i(x)$ is piecewise linear with a single kink at K_i, the portfolio payoff $\hat{g}(x)$ is a piecewise linear function with kinks at the strikes $\{K_i : w_i \neq 0\}$. The **approximation quality** is therefore fundamentally limited by the density and placement of available strikes relative to the curvature of $g^*(x)$.

The general discretization problem is to minimize the P-weighted squared error between $g^*(x)$ and $\hat{g}(x)$ subject to the budget constraint. This forms a convex quadratic program. However, each regime's optimal payoff $g^*(x)$ has a characteristic *shape* that admits a more targeted discretization strategy.

15.2 Fractional vs. Integer Contracts

The nature of the discretization problem changes fundamentally depending on whether fractional option contracts are permitted.

15.2.1 Case 1: Fractional Contracts Enabled ($\mathbf{w} \in \mathbb{R}^N$)

When fractional contracts are allowed, the weights can be any real number. The portfolio can be arbitrarily scaled to perfectly match the budget without distorting the shape. The algorithms compute a "raw" analytical weight vector $\mathbf{w}_{raw}$, compute its market cost C_{raw}, and then apply a continuous scaling factor $\alpha = K/C_{raw}$ to obtain the final executed weights $\mathbf{w}_{final} = \alpha \mathbf{w}_{raw}$.

15.2.2 Case 2: Fractional Contracts Disabled ($\mathbf{w} \in \mathbb{Z}^N$)

When contracts must be whole integers (e.g., 1 contract = 100 shares), the problem becomes a much harder **Integer Programming** problem. Independent rounding destroys the structural integrity of the spreads.

To handle integer constraints, the approaches diverge based on the regime type:

- **Fixed-Structure Regimes (Sniper, Bunker):** These rely on specific, delicate structures (Butterflies and Box Spreads) to eliminate tail risk. The spread must be treated as a single indivisible "Unit" with a base integer ratio. The maximum number of whole units the user can afford is $n = \lfloor K/C_{unit} \rfloor$. The final portfolio is the base integer ratio multiplied by n, with any leftover budget remaining in cash.
- **Smooth-Curve Regimes (Insurer, Farmer, Sentinel):** These use Carr-Madan replication, resulting in dozens of legs with varying weights. If the user's budget is small, they may not be able to afford even 1 contract of each required leg. This creates a **Sparsity vs. Budget** problem. To solve this, we employ **Adaptive Grid Coarsening (Strike Thinning)**:
  1. Calculate the continuous Carr-Madan weights $\mathbf{w}_{ideal}$ using all N strikes.
  2. Calculate the cost to execute this exact structure at a minimum of 1 unit: $C_{min} = \sum \max(1, \lceil |w_{ideal,i}| \rceil) \cdot p_i^{mkt}$.
  3. While $C_{min} > K$ (budget is too small), find the strike K_{drop} with the lowest "importance" (curvature $\times$ probability: $|\Delta_j^2 g^*| \cdot P(K_j)$).
  4. Remove K_{drop} from the available strikes and re-calculate the Carr-Madan weights on the remaining $N-1$ strikes (effectively smoothing out the curve to require fewer bends/options).
  5. Repeat until $C_{min} \leq K$. The result is a sparse, integer-feasible strategy that perfectly fits the budget and is mathematically guaranteed to be the best piecewise linear approximation on the reduced strike grid.

15.3 Regime 1: The Sniper

15.3.1 Case 1: Fractional Contracts Enabled

Theoretical Approximation. The Sniper's optimal payoff is a Dirac delta at $x^* = \text{argmax}_x \Lambda(x)$, concentrating all capital at the single most mispriced price point. A Dirac delta is not tradeable; its finite-width approximation in option space is the **Butterfly Spread**. A butterfly centered at strike K_c with wings at $K_c - \delta$ and $K_c + \delta$ produces a triangular payoff that converges to a Dirac delta as $\delta \to 0$.

Algorithm.

1. Compute $x^* = \text{argmax}_x \Lambda(x)$ from the spline representations of $P(x)$ and $Q(x)$.
2. Find the available strike K_c nearest to x^*.

3. Select the tightest available wing strikes $K_L < K_c < K_R$ (ideally the immediate neighbors in the strike grid).

4. Construct the raw butterfly vector $\mathbf{w}_{raw}$: Long 1 call at K_L, Short $\frac{K_R-K_L}{K_c-K_L}$ calls at K_c, Long $\frac{K_c-K_L}{K_R-K_L} \cdot \frac{K_R-K_c}{K_c-K_L}$ calls at K_R.

5. Apply continuous scaling $\alpha = K/C_{raw}$ so the final cost exactly matches the budget.

15.3.2 Case 2: Fractional Contracts Disabled

Theoretical Approximation. A discrete butterfly spread relies on a precise internal ratio to perfectly cancel tail exposure. Independent rounding of individual leg weights is forbidden. The butterfly must be treated as a single, indivisible "Unit."

Algorithm.

1. Execute Steps 1–4 of Case 1 to identify the optimal wing strikes and raw integer ratio.

2. Calculate the market cost C_{unit} of exactly one unit of this minimal integer butterfly.

3. The maximum number of whole units the user can afford is $n = \lfloor K/C_{unit} \rfloor$.

4. The final portfolio is the base integer ratio multiplied by n. Leftover budget remains in cash.

15.4 Regime 2: The Insurer + Sniper

15.4.1 Case 1: Fractional Contracts Enabled

Theoretical Approximation. The Insurer's safety portfolio $g_{safe}(x) = \tilde{T}(x)$ is a smooth, non-linear curve. The best piecewise linear approximation is obtained by interpolating g_{safe} at the available strikes $K_1 < K_2 < \cdots < K_N$. Any piecewise linear function with kinks at $\{K_j\}$ can be decomposed into a portfolio of calls and puts via the **Carr-Madan replication formula**:

$$\hat{g}(x) = \hat{g}(K_1) + \hat{g}'(K_1^+)(x - K_1)^+ + \sum_{j=2}^{N} \Delta_j^2 \hat{g} \cdot (x - K_j)^+ \tag{43}$$

where $\Delta_j^2 \hat{g}$ is the discrete second difference (change in slope at K_j). Each term corresponds to a call position at strike K_j with weight equal to the slope change.

Algorithm.

1. Compute $g_{safe}(x) = \tilde{T}(x)$ using the spline representations.

2. Evaluate $g_{safe}(K_j)$ at each available strike K_j.

3. Compute the discrete second differences $\Delta_j^2 g_{safe}$ at each interior strike.

4. Construct the raw replicating portfolio $\mathbf{w}_{raw}$: each non-zero $\Delta_j^2 g_{safe}$ becomes a call position at K_j.

5. Allocate any remaining budget to the Sniper butterfly at $x^* = \operatorname{argmax} \Lambda$, returning the full fractional portfolio.

15.4.2 Case 2: Fractional Contracts Disabled

Theoretical Approximation. The Carr-Madan formula typically assigns non-zero weights to every interior strike. If the budget K is small, we must solve the Sparsity vs. Budget problem by applying Adaptive Grid Coarsening (Strike Thinning).

Algorithm.

1. Compute the continuous Carr-Madan weights $\mathbf{w}_{ideal}$ using all N strikes.
2. Check integer minimum cost: $C_{min} = \sum \max(1, \lceil |w_{ideal,j}| \rceil) \cdot p_j^{mkt}$.
3. While $C_{min} > K$:
   (a) Identify the lowest importance strike: $K_{drop} = \text{argmin}_j \left(|\Delta_j^2 g_{safe}| \cdot P(K_j) \right)$.
   (b) Remove K_{drop} and re-evaluate Carr-Madan weights on the remaining strikes.
   (c) Recalculate C_{min}.
4. Once $C_{min} \leq K$, output the surviving sparse integer weights. Leftover budget goes to an integer Sniper unit.

15.5 Regime 3: The Farmer

15.5.1 Case 1: Fractional Contracts Enabled

Theoretical Approximation. The Farmer's optimal payoff $g^*(x) = \max(\tilde{T}(x), \frac{1}{\lambda}\Lambda(x))$ features a flat floor region and a curved growth region. The interpolation-then-replication strategy from Regime 2 applies identically. The error is bounded by the curvature of the likelihood ratio $\Lambda''(x)$. Because the logarithmic objective $\mathbb{E}_P[\ln(g)]$ diverges to $-\infty$ if $g = 0$, the interpolant must be carefully managed to remain strictly positive.

Algorithm.

1. Solve for λ via Newton-Raphson to obtain the continuous $g^*(x)$.
2. Evaluate $g^*(K_j) = \max(\tilde{T}(K_j), \frac{1}{\lambda}\Lambda(K_j))$ at each available strike.
3. Compute discrete second differences and construct the raw replicating portfolio via Carr-Madan.
4. If the raw discrete portfolio cost diverges from the budget, adjust λ and iterate until convergence.

15.5.2 Case 2: Fractional Contracts Disabled

Theoretical Approximation. As in Regime 2, the budget constraint forces a sparse approximation of the ideal curve. We apply Adaptive Grid Coarsening to reduce the number of required option legs while maintaining the best possible piecewise linear fit.

Algorithm.

1. Execute the outer Newton loop from Case 1 to find the optimal λ and the continuous Carr-Madan weights $\mathbf{w}_{ideal}$.
2. Check integer minimum cost: $C_{min} = \sum \max(1, \lceil |w_{ideal,j}| \rceil) \cdot p_j^{mkt}$.

3. While $C_{min} > K$:
   (a) Identify $K_{drop} = \text{argmin}_j \left(|\Delta_j^2 g^*| \cdot P(K_j) \right)$.
   (b) Remove K_{drop} and re-evaluate Carr-Madan weights on the remaining strikes.
   (c) Recalculate C_{min}.
4. Once $C_{min} \leq K$, output the surviving sparse integer weights. Leftover budget remains in cash.

15.6 Regime 4: The Bunker

15.6.1 Case 1: Fractional Contracts Enabled

Theoretical Approximation. The Bunker's constant payoff $g^*(x) = M$ is exactly replicable over all $x \in [0, \infty)$ using a **Box Spread** $[K_i, K_j]$. Because all four legs (Long Call K_i, Short Call K_j, Long Put K_j, Short Put K_i) are included, the payoff is perfectly flat (slope zero) both inside and outside the $[K_i, K_j]$ interval. The short legs perfectly cancel the tail growth of the long legs. This guarantees the absolute minimum floor M without any tail risk.

Algorithm. Because of market pricing inefficiencies, the widest possible box spread is not necessarily the most capital-efficient. To maximize the floor M, we must search all possible pairs of strikes:

1. Iterate through all valid pairs of available strikes (K_i, K_j) where $K_i < K_j$.
2. Construct the raw Box Spread $\mathbf{w}_{raw}$: $+1$ Call at K_i, -1 Call at K_j, $+1$ Put at K_j, -1 Put at K_i.
3. Scale the continuous position $\alpha = K/C_{raw}$ so the total cost exactly equals the budget K. The achieved floor is $M_{achieved} = \alpha \cdot (K_j - K_i)$.
4. Select the pair (K_i, K_j) that maximizes $M_{achieved}$. This $\mathcal{O}(N^2)$ search is computationally trivial.

15.6.2 Case 2: Fractional Contracts Disabled

Theoretical Approximation. Like the Sniper, the Box Spread is a fixed-structure regime. The 4-leg ratio $(+1, -1, +1, -1)$ must be strictly maintained to guarantee the flat tail payoff. It must be treated as a single indivisible unit.

Algorithm.

1. Iterate through all valid pairs of available strikes (K_i, K_j) where $K_i < K_j$.
2. Calculate the market cost C_{unit} of exactly one integer unit of the Box Spread.
3. The maximum number of whole units the user can afford is $n = \lfloor K/C_{unit} \rfloor$.
4. The achieved floor for this pair is $M_{achieved} = n \cdot (K_j - K_i)$.
5. Select the pair (K_i, K_j) that maximizes $M_{achieved}$. Leftover budget remains in cash.

15.7 Regime 5: The Sentinel

15.7.1 Case 1: Fractional Contracts Enabled

Theoretical Approximation. The Sentinel's hybrid payoff features a sacrifice zone (where g can drop below M), a protected floor M, and a Farmer growth region. Setting $g(x) = 0$ in the sacrifice zone may not align with available strikes. The practical solution is to snap the sacrifice zone boundary to the nearest available strike K_s, leaving the tail unhedged while protecting the region $x > K_s$.

Algorithm.

1. Compute the continuous sacrifice zone based on the p^* VaR constraint.
2. Snap the boundary to the nearest available strike K_s and verify the discrete P-mass does not exceed p^*.
3. On the protected strikes $\{K_s, \ldots, K_N\}$, evaluate $g^*(K_j) = \max(M, \frac{1}{\lambda}\Lambda(K_j))$.
4. Compute discrete second differences and construct the raw replicating portfolio.
5. Solve for M and λ jointly (VaR and budget constraints) by applying continuous scaling and iterating the discrete portfolio construction.

15.7.2 Case 2: Fractional Contracts Disabled

Theoretical Approximation. The Sentinel combines the complexity of the Farmer's curved growth with the boundary conditions of the sacrifice zone. When restricted to integers, we must apply Adaptive Grid Coarsening to the protected strikes $\{K_s, \ldots, K_N\}$ to find a sparse portfolio that fits the budget.

Algorithm.

1. Execute the continuous optimization (Steps 1–5 of Case 1) to find the optimal M, λ, boundary K_s, and continuous Carr-Madan weights $\mathbf{w}_{ideal}$ on the protected strikes.
2. Check integer minimum cost: $C_{min} = \sum_{j \geq s} \max(1, \lceil |w_{ideal,j}| \rceil) \cdot p_j^{mkt}$.
3. While $C_{min} > K$:
   (a) Identify $K_{drop} = \text{argmin}_{j>s} \left(|\Delta_j^2 g^*| \cdot P(K_j) \right)$. (Do not drop the boundary strike K_s).
   (b) Remove K_{drop} and re-evaluate Carr-Madan weights on the remaining protected strikes.
   (c) Recalculate C_{min}.
4. Once $C_{min} \leq K$, output the surviving sparse integer weights. Leftover budget remains in cash.

15.8 Complexity of Discrete Approximation

Regime	**Analytical g^* Computation**	**Discretization (Fractional)**	**Discretization (Integer)**
1: Sniper	$\mathcal{O}(M)$	$\mathcal{O}(\log N)$	$\mathcal{O}(\log N)$
2: Insurer + Sniper	$\mathcal{O}(M \log M)$	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$
3: Farmer	$\mathcal{O}(L \cdot M)$	$\mathcal{O}(L \cdot N)$	$\mathcal{O}(L \cdot N + N^2)$
4: Bunker	$\mathcal{O}(1)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$
5: Sentinel	$\mathcal{O}(L \cdot M)$	$\mathcal{O}(L \cdot N)$	$\mathcal{O}(L \cdot N + N^2)$

The runtime of the discretization step is dominated by the analytical computation of g^*, resulting in total end-to-end runtimes of $\mathcal{O}(M \log M)$ or better (where M is the spline grid resolution). When fractional contracts are disabled, the smooth-curve regimes incur an additional $\mathcal{O}(N^2)$ penalty due to the Adaptive Grid Coarsening (Strike Thinning) algorithm. However, because the number of strikes N is small, this adds negligible overhead. By contrast, search-based optimization (like GSGD) operates entirely in discrete space with exponential or highly bounded local search runtime $\mathcal{O}(K_{seeds} \cdot I \cdot d)$. Analytical construction followed by discretization is approximately $50\times$ faster than GSGD and guarantees finding the global optimum (subject only to discretization error at the strikes).

Part V
Reference

16 Glossary

- **PDF** – Probability Density Function. Describes the likelihood of each possible stock price.
- **CDF** – Cumulative Distribution Function. $\Psi(x) = \int_{-\infty}^{x} \mu(y)\, dy$.
- **IV** – Implied Volatility. The market's expectation of future volatility, back-solved from option prices via Black-Scholes.
- **Strike Price (K)** – The price at which an option can be exercised.
- **Premium (p)** – The market price of an option contract.
- **Spread** – A combination of two or more option legs.
- **CAGR** – Compound Annual Growth Rate. The annualized return assuming reinvestment.
- **Kelly Criterion** – The optimal fraction of capital to wager for maximum geometric growth.
- **PEE** – Probabilistic Early Expiration. A backend caching strategy that refreshes data before it strictly expires.