Robot Baseball: Reaching a Full Count under Optimal Play

Saxon Lee

2nd of October 2025

Abstract

Jane Street dropped a game theory puzzle, so I gave it a whirl. I set up the puzzle as a small stochastic zero-sum game. For a fixed home–run parameter p, I firstly solved the pitcher–batter stage game at every count for the *expected score*. From those equilibrium values I extracted the mixed strategies. Holding those mixes fixed, I then computed the probability that the at-bat ever hits the full-count state $(3, 2)$. To bring it all home, I searched over $p \in [0, 1]$ to maximise that probability. The peak probability is

$$\boxed{q_{\max} = 0.2959679934} \quad \text{at} \quad \boxed{p \approx 0.2269732296}.$$

This was the output of my runs to 10 decimal places for q, which resulted in the correct submission.

1 How I modeled the at-bat

Each at-bat is a sequence of pitches. The state is the count (b, s) with $b \in \{0, 1, 2, 3, 4\}$ balls and $s \in \{0, 1, 2, 3\}$ strikes. Walk at $b = 4$ pays 1. Strikeout at $s = 3$ pays 0. A swing at a strike yields a home run with probability p for a payoff 4, else it behaves like a strike and the count increments. Both players pick their actions at the same time on each pitch: the pitcher picks {Ball, Strike}, the batter picks {Wait, Swing}. Both aim to optimise *expected score*.

Let $V(b, s)$ be the batter's optimal expected score from (b, s). Terminal values are $V(4, s) = 1$ and $V(b, 3) = 0$.

2 The per-pitch 2×2 game

At a non-terminal state (b, s), write the 2×2 payoff matrix M for the batter as

$$M = \begin{pmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{pmatrix} = \begin{pmatrix} V(b{+}1, s) & V(b, s{+}1) \\ V(b, s{+}1) & p \cdot 4 + (1-p)\, V(b, s{+}1) \end{pmatrix}.$$

Rows are pitcher actions $\{\text{Ball}, \text{Strike}\}$. Columns are batter actions $\{\text{Wait}, \text{Swing}\}$. Note the off-diagonal symmetry: $M_{01} = M_{10} = V(b, s{+}1)$. This symmetry matters: it forces the two players' interior mixes to match at each state. I use that below.

Stage-game value and mixes. Define

$$D \;=\; M_{00} - M_{01} - M_{10} + M_{11} \;=\; V(b{+}1, s) - V(b, s{+}1) - V(b, s{+}1) + \big(p \cdot 4 + (1-p) V(b, s{+}1)\big).$$

When the saddle is interior ($0 < x < 1$, $0 < y < 1$), the equilibrium mixes and the value are

$$x \;=\; \frac{M_{11} - M_{10}}{D}, \qquad y \;=\; \frac{M_{11} - M_{01}}{D}, \qquad V(b, s) \;=\; \frac{M_{00} M_{11} - M_{01} M_{10}}{D}.$$

Because $M_{01} = M_{10}$, we get $x = y$ in all interior states, which I also check numerically. When the computed mix lands outside $[0, 1]$, I fall back to the appropriate pure strategy and take the max–min value.

3 Solving V for a fixed p

I run value iteration on the 12 non-terminal states $(b, s) \in \{0, 1, 2, 3\} \times \{0, 1, 2\}$. On each sweep:

1. Build the local matrix M from the current guess V.
2. Compute the stage-game saddle value (and, if needed, the pure endpoints).
3. Update $V(b, s)$ by that value.

The process converges fast with a tight tolerance (I used 10^{-14}). This yields the fixed point $V(\cdot, \cdot)$ for that p.

4 From expected score to the full-count probability

Once V converges, I compute the equilibrium mixes $(x_{b,s}, y_{b,s})$ at each state using the same 2×2 formulas. These mixes induce a Markov chain over counts. I then compute the probability $q(b, s)$ that the chain ever hits the full-count state $(3, 2)$ before absorption elsewhere.

With mixes fixed at (b, s),

$$\begin{aligned} P(\text{Ball,Wait}) &= x\,y, \\ P(\text{Ball,Swing}) &= x(1-y) \quad \text{(counts as a strike)}, \\ P(\text{Strike,Wait}) &= (1-x)y \quad \text{(strike)}, \\ P(\text{Strike,Swing,HR}) &= (1-x)(1-y)\,p, \\ P(\text{Strike,Swing,no HR}) &= (1-x)(1-y)(1-p) \quad \text{(strike)}. \end{aligned}$$

Let $q(b, s)$ be the hit–$(3, 2)$ probability from (b, s). With the boundary $q(3, 2) = 1,\ q(4, s) = 0,\ q(b, 3) = 0$, the linear equations are

$$q(b, s) = P(\text{Ball,Wait})\, q(b{+}1, s) + \big[P(\text{Ball,Swing}){+}P(\text{Strike,Wait}){+}P(\text{Strike,Swing,no HR})\big]\, q(b, s{+}1).$$

I treat $(3, 2)$ as absorbing for the purpose of this computation and solve the resulting linear system in one shot. The value we want is $q(0, 0)$. index=5

5 Maximizing over p

I view $q(0, 0)$ as a function $q(p)$. I first scan a coarse grid on $[0, 1]$ to bracket the peak, then refine with a golden-section search. The function is smooth and unimodal in practice. The maximum I find is

$$\boxed{q_{\max} = 0.2959679934} \quad \text{at} \quad \boxed{p \approx 0.2269732296}.$$

Reruns with stricter tolerances and a denser grid agree to the shown digits.

6 What I tried first and why I changed course

My first pass tried to make the players optimise the chance of *reaching* full count directly. That was a mistake. The puzzle states both players optimise *expected score*. So the equilibrium I need is with respect to score, not path probability. Once I fixed the objective and did the two-stage flow (solve score game $\rightarrow$ compute hit probability), the numbers made sense and were stable.

A second early slip was to overcomplicate the 2×2 saddle with a linear program. The closed forms are simple and robust here, and the off-diagonal symmetry $M_{01} = M_{10}$ helps: in interior states it implies $x = y$. I still keep endpoint checks for safety.

7 Sanity checks

I checked the extremes. When $p = 0$, swinging at strikes has no upside, so the pitcher leans on strikes and the full count is rare. When p gets large, the pitcher avoids serving strikes you can hit, which also pulls the full-count chance down. So a single peak inside $(0, 1)$ is reasonable, and the search finds it near $p \approx 0.227$. I also verified that recomputing the absorption probability at that p with independent code paths gives the same q up to 10^{-9}.

8 Result

$$\boxed{q_{\max} = 0.2959679934 \;\text{ at }\; p \approx 0.2269732296.}$$

This is the maximal probability that an at-bat reaches a full count under optimal play by both sides, to ten decimal places for q.

Reproduce it

Solve V by value iteration using the 2×2 saddle formulas above, extract mixes, solve the linear system for the hit probability to $(3, 2)$, and maximise $q(0, 0)$ over p. A grid + golden search is enough. Tight tolerances only matter for the last few decimals.