How Accurate Are Open-Source Social Security Calculators?

Shirshak Jung Thapa
Independent Researcher

Abstract

People use open-source calculators to estimate their Social Security retirement benefit, and no study we know of checks them against an exact answer. We built the Social Security Administration's published calculator, ran 1,231 synthetic (made-up) workers through it, many on the edges of the rules, and used its results as the answer key. A second program, written from the law, agrees on every worker. Of five calculators, one matches the agency for every worker. Three give a wrong monthly benefit for 2, 15 and 19 percent of workers, mostly through three small mistakes. Two round to the nearest cent before rounding down, making some benefits a dollar too high. Two miss that the law counts someone born on 1 January as born the year before, worth up to $175 a month. One ignores the yearly limit on countable earnings, putting a high earner typically $597 a month too high. An ordinary worker, born mid-month and earning under the limit, gets the right amount or a dollar more. The fifth counts only earnings from age 22 to 61, as its code states, and runs a little low. Why were the mistakes not caught? Developers test against published worked examples, and three, in a Congressional Research Service report and the federal regulations, print an amount the rules do not give. Checks written without an answer key caught the two largest mistakes and missed the rounding one. We release the workers, official answers and checks so anyone can test a calculator this way.

Keywords: Social Security benefits; open-source calculators; differential testing; metamorphic testing; mutation testing; oracle problem

1. Introduction

Take a worker born on 1 January 1960 who earned $40,000 in every year from 1990 to 2025 and claims a retirement benefit for January 2026. The Social Security Administration's own calculator puts the benefit at $2,619 a month. One open-source calculator says $2,538. Another says $2,508. Move the birthday one day later, to 2 January, and all three agree on $2,493.

The reason is one sentence in the regulations: a person reaches an age on the day before the birthday (20 CFR 404.102). Someone born on 1 January 1960 turned 62 on 31 December 2021, so the law treats that person as born in 1959. The worker is therefore first eligible in 2021, not 2022, and receives the yearly increase of December 2021 (5.9 percent). The full retirement age is 66 and 10 months, not 67. The first calculator had the full retirement age right but placed eligibility in 2022; it left out the 2021 increase and came out $81 a month too low. The second used the calendar year 1960 for both steps and came out $111 too low.

Small rules like this one are where benefit calculators go wrong. The method for working out a retirement benefit is public, and its main steps are well known. The fine details are less well known: at which steps an amount is rounded, and whether it is rounded down or to the nearest value; on which day a person

reaches an age; and which year's published figures apply to a given worker. Free calculators, many of them open source (their code is public and anyone may reuse it), help people decide when to claim, a choice that sets the size of their monthly benefit for life (Shoven and Slavov, 2014). This paper asks how closely such calculators follow the rules.

Answering that needs an exact answer key, and one exists. The agency publishes the source code of its Detailed Calculator (Social Security Administration, 2026a). It is a public calculator and not the system that decides claims, but it is the agency's own statement of the method in code. We built it unchanged and ran it on 1,231 synthetic workers: earnings histories we generated, with 331 of them placed where a rule changes, such as a birthday on the first of a month or earnings exactly at the yearly limit. To test the key itself, we wrote a second program from the law and regulations alone. The two agree on every worker at three stages of the calculation. "Accurate" in this paper therefore has a precise meaning: a calculator agrees with the agency's program, to the dollar, on these workers. It measures a calculator against the rules, not how often real claimants would see a wrong figure.

We tested five independent open-source calculators. One, PolicyEngine-US, matches the agency's program on all 1,231 workers. Three give a wrong monthly benefit for some workers: ssa.tools for 26, the npm package social-security-calculator for 186, and retiretui for 230. The fifth, the RetireGolden engine, does so for 968, because of two simplifications its own source code states: it counts only earnings from age 22 to 61, and it rounds each adjusted year of earnings down to the dollar.

All but one of the differences trace to a small number of nameable rules, and two of the mistakes recur in projects that share no code. One is the birthday rule above, which moves the benefit by tens of dollars and sometimes by more than a hundred. The other is the order of rounding: the law rounds certain amounts down, and two calculators first round to the nearest cent, which adds at most one dollar a month. In one calculator, earnings above the yearly taxable limit are counted in full, which overstates the benefit by up to about $800 a month; its documentation does not say whether earnings should be entered already cut to the limit. We reported the differences to the three projects (ssa.tools issue 637, social-security-calculator issue 44 and retiretui issue 104).

Why do these mistakes go unnoticed? Without the agency's program, a developer tests against published worked examples and against checks of their own. We examined both. Three published examples, two in a Congressional Research Service report and one in the regulations themselves, print an amount that a correct program does not return. As for a developer's own checks, we wrote four families of them. They catch 256 of 274 planted mistakes, and the 18 they miss include the kinds of mistake the calculators make. Run on the calculators themselves, the checks catch the two largest mistakes and miss the rounding one, and one calculator that rounds the wrong way returns the very amounts that a published example prints wrongly.

We also ran a pilot with an AI model. The model was given the official documents and nothing else, and was asked to do two things, five times each. The first was to write a program that works out a worker's monthly benefit from the date of birth, the month of the claim and the earnings of each year. All five programs match the agency's calculator on every worker tested. The second was to find in the documents the 165 yearly figures such a program relies on, such as the national average wage for each year and the

full retirement age for each year of birth. Three of the five tables are right in every entry. The other two each have one wrong number, and our checks caught both. The pilot is one model and ten runs, which is enough to show that such mistakes happen but not how often.

This paper makes four contributions.

1. **An answer key.** A set of 1,231 workers with the agency's result at three stages, confirmed by a second program written from the law alone.
2. **A measurement.** As far as we know, the first comparison of independently written open-source Social Security calculators with the agency's own program, to the dollar, with every difference but one traced to a named rule.
3. **An explanation.** Evidence of why the mistakes go unnoticed: published worked examples that print an amount the rules do not give, and checks written without an answer key that miss the rounding mistake.
4. **A package.** The workers, the official results, the checks and the programs that run each calculator, released so that any calculator can be tested the same way (Thapa, 2026; https://doi.org/10.5281/zenodo.24375957).

The study covers retired workers born from 1959 to 1964 who claim in 2026 at or before full retirement age; it does not cover disability, family or survivor benefits. Section 2 explains how a benefit is computed, and Section 3 describes the answer key. Sections 4, 5 and 6 report the calculators, the published examples and the checks. Section 7 gives the pilot, Section 8 advice for someone building a calculator, Section 9 the limits and Section 10 related work.

2. Background: how a retirement benefit is computed

A retirement benefit is not computed in one go. It is worked out in steps, and the result of each step is the starting point of the next. Three results along the way have official names. The first is the average indexed monthly earnings: the worker's average monthly pay over the 35 best years, after older earnings are raised to reflect the growth in wages. The second is the primary insurance amount: the monthly benefit the worker would get by claiming at full retirement age. The third is the monthly benefit itself: what the worker actually receives, which is lower if the claim is made early. This paper compares calculators at all three, so that a mistake can be traced to the step where it happens. We follow one worker through the steps. She was born on 2 January 1964, earned $30,000 in every year from 1990 to 2025, and claims for January 2026.

When she becomes eligible. The earliest age at which a retirement benefit can start is 62. In Social Security law a person reaches an age on the day before the birthday (20 CFR 404.102). Our worker's 62nd birthday is 2 January 2026, so she is 62 from 1 January 2026. The year in which a worker reaches 62 is called the year of first eligibility. For her it is 2026, and that year decides which of the agency's yearly figures are used in her calculation. One more rule applies to anyone who claims before full retirement age: a benefit is paid only for a month in which the person is 62 on every day of the month (42 U.S.C. 402(a)).

She is 62 from the first day of January 2026, so January 2026 is the first month she can be paid for. Had she been born one day later, on 3 January, her first month would be February.

Which earnings count. Social Security tax is charged only on earnings up to a yearly limit, called the taxable maximum, and only earnings up to that limit count toward the benefit (42 U.S.C. 415(e)(1), 430). The limit rises in most years: it was $51,300 in 1990 and $176,100 in 2025. A worker who earned $200,000 in 2025 is treated as having earned $176,100. Our worker's $30,000 is under the limit in every year, so all of it counts.

Adjusting for wage growth. Wages were much lower decades ago, so $30,000 in 1990 was a far larger share of a typical wage than $30,000 is today. To put all years on the same footing, older earnings are raised in line with the growth in average wages. The agency publishes the national average wage for each year, called the average wage index. Earnings are brought up to the wage level of one reference year: the second year before the year of first eligibility, which for her is 2024 (42 U.S.C. 415(b)(3)). The average wage was $21,027.98 in 1990 and $69,846.57 in 2024. Dividing the second by the first shows that wages grew about 3.32 times over. Her $30,000 from 1990 is multiplied by that same factor, which gives $99,648.05. This is the amount used for 1990 in the rest of the calculation, in place of the original $30,000. Each adjusted amount is rounded to the nearest cent (20 CFR 404.211(d)(3)). Earnings in the reference year and after are not adjusted, so her 2024 and 2025 earnings stay at $30,000.

The average. Only the 35 years with the highest adjusted earnings are used. A worker with fewer than 35 years of earnings still has 35 years counted, with zero for each missing year (42 U.S.C. 415(b)(2)). Our worker has 36 years of earnings, 1990 to 2025, so one year is left out. Her lowest years after adjustment are 2024 and 2025, at $30,000 each, and one of them is dropped. The remaining 35 years add up to $2,038,205.85. That total is divided by 420, the number of months in 35 years, to give an average per month: 4,852.87. The law rounds this down to the whole dollar (42 U.S.C. 415(b)(1), (e)(2)). Her average indexed monthly earnings, the first of the three named amounts, are $4,852.

The formula. The next step turns the monthly average into the primary insurance amount. This is the official name for the monthly benefit a worker is paid if the claim is made exactly at full retirement age, neither early nor late (42 U.S.C. 402(a), 415(a)). The benefit she is actually paid is worked out from it in the last step.

The formula cuts the monthly average into three slices and pays a different share of each: 90 percent of the first slice, 32 percent of the second and 15 percent of the third (42 U.S.C. 415(a)(1)(A)). A low earner therefore gets back a larger share of earnings than a high earner. The two dollar amounts where one slice ends and the next begins are called bend points. The agency sets them each year, and a worker keeps the bend points of the year of first eligibility. For 2026 they are $1,286 and $7,749.

Her monthly average is $4,852, and it falls into the slices like this:

- the first $1,286, paid at 90 percent: 1,157.40;
- the next $3,566, which takes her from $1,286 up to her average of $4,852, paid at 32 percent: 1,141.12;
- nothing above $7,749, so the 15 percent slice adds nothing.

The two results add up to 2,298.52. The law rounds this down to the next ten cents. Her primary insurance amount, the second of the three named amounts, is $2,298.50.

Yearly increases. Benefits are raised each year to keep up with prices. The raise is called the cost-of-living increase. It is announced as a percentage and takes effect each December. A worker's primary insurance amount receives every increase from the year of first eligibility onward, whether or not the worker has claimed yet. After each increase the result is rounded down to the next ten cents (42 U.S.C. 415(i)(2)(A)(ii)). Our worker becomes eligible in 2026 and claims for January 2026, before the first December that applies to her, so her amount stays at $2,298.50. A worker who became eligible in 2023 would by January 2026 have had three increases, of 3.2, 2.5 and 2.8 percent, with a rounding down after each one. This repeated rounding is where two of the calculators in Section 4 go wrong.

Claiming early. The last step adjusts for the age at which the worker claims. Full retirement age is the age at which the full primary insurance amount is paid. It is 67 for people born in 1960 or later, and 66 and 10 months for people born in 1959 (42 U.S.C. 416(l)). A worker may claim as early as 62, but the benefit is then lowered for good. The reduction is counted in months: five-ninths of one percent for each of the first 36 months before full retirement age, and five-twelfths of one percent for each month beyond 36 (42 U.S.C. 402(q)). Our worker reaches 67 on 1 January 2031, so a claim for January 2026 is 60 months early. The first 36 months take off 20 percent and the other 24 months take off 10 percent, 30 percent in all. She keeps 70 percent of $2,298.50, which is 1,608.95. The law rounds the monthly benefit down to the whole dollar (42 U.S.C. 415(g)). She receives $1,608 a month, the third of the three named amounts.

Table 1 sets the steps side by side, with her figure at each step and the rounding the law requires there.

Table 1. The worker of this section, step by step.

Step	Her figure	Rounding the law sets
Year of first eligibility	2026 (reaches 62 on 1 January 2026)	none
Reference year for wage growth	2024	none
1990 earnings after adjustment	$99,648.05	to the nearest cent
Total of the highest 35 years	$2,038,205.85	none
Average indexed monthly earnings	$4,852 (from 4,852.87)	down to the dollar
Primary insurance amount	$2,298.50 (from 2,298.52)	down to ten cents
After yearly increases	$2,298.50 (none yet)	down to ten cents, each year
Monthly benefit, 60 months early	$1,608 (from 1,608.95)	down to the dollar

We ran this worker through the agency's calculator. It returns the same three amounts: $4,852, $2,298.50 and $1,608.

Two things in these steps matter for the rest of the paper.

The first is the direction of rounding. At four steps the law rounds down, never to the nearest value: the monthly average, the primary insurance amount, each yearly increase, and the monthly benefit (Table 1). An amount of 2,298.59 becomes 2,298.50, not 2,298.60. A program that rounds to the nearest value will be right for most workers and ten cents or a dollar off for some. So will a program that first rounds to the

cent and then rounds down: an exact result of 2,272.4968 becomes 2,272.50 at the cent, which leaves nothing to round down, where the law gives 2,272.40.

The second is the rule that a person reaches an age on the day before the birthday. It decides three things: the year of first eligibility, the full retirement age, and how many months early a claim is. For most birthdays the rule makes no difference. It matters for a person born on the first day of a month, who reaches every age in the month before, and most of all for a person born on 1 January, who reaches every age in the year before.

The calculation also relies on four sets of figures that the agency publishes every year: the average wage index, the taxable maximum, the cost-of-living increase and the bend points. A calculator can store these figures exactly as the agency publishes them, or it can work some of them out with a formula. The two do not always agree. Section 4 shows a calculator that works out the taxable maximum by formula and, for some years, gets a figure different from the published one.

3. The answer key

3.1 The agency's calculator

The Social Security Administration publishes a benefit calculator of its own, called the Detailed Calculator, together with its source code, the instructions written by programmers from which the program is built (Social Security Administration, 2026a). We used version 2026.2, released on 13 July 2026. Because it is the agency's own statement, in code, of how a benefit is worked out, it is the most authoritative answer key publicly available. The agency notes that it is not the same calculator as the one used for official calculations and may produce results that differ from an official one. The approximations it lists all lie outside this study: benefits from before 1965, a frozen minimum benefit, disability, and benefits for dependents and survivors. The agency adds that in its experience the calculator "matches the official calculations very well", and that most differences come from earnings added late to a worker's record or from different assumptions about the future. Neither arises here, where every earnings history is fixed and every claim falls in 2026.

The source code includes a program made for running many cases at once. It reads a text file in which each case gives a date of birth, the month in which the benefit starts and a list of yearly earnings, and it writes one line of results for each case. A setting chosen when the program is built adds the average indexed monthly earnings to that line. With it, we get all three named amounts of Section 2 for every worker.

Turning source code into a program that runs is called building it. We built this program on Linux with a current compiler, the tool that does the building (g++ 15.2). All 230 files of the agency's source code were used exactly as published; we changed nothing in them. Three small additions were needed to make the build work on Linux, and none of them takes part in computing a benefit: current copies of two supporting files, taken unchanged from the Windows folder of the same download; an instruction to the compiler to include a standard part of the language; and a simple replacement for two text functions that exist only on Windows and serve a printed report this study does not use. The build script is in the release.

A program we built ourselves could differ from the one the agency intends, if something went wrong in the building. To rule that out, we compared it with the version the agency builds itself and offers for download as a ready-made Windows program. We ran both on the same cases: the 1,231 workers of this study and the 30 sample cases the agency supplies with the program, which include disability and survivor benefits and workers from before 1979. The two programs wrote 1,263 lines of results, and all 1,263 are identical, character for character, apart from the extra columns that our setting adds.

Version 2026.2 holds the published yearly figures that a worker who reaches 62 in 2026 needs. It does not hold the cost-of-living increase that was announced in October 2026 and takes effect in December 2026, because that increase was not yet known when the version came out. To stay within what the program knows, every claim in this study starts between January and November 2026.

3.2 A second program, written from the rules

We did not want the answer key to rest on one program alone. We might have given the agency's program its input in the wrong form, or misread what its output means. So we wrote a second program of our own, working only from the law and the regulations cited in Section 2 and without reading the part of the agency's code that does the calculation. When two programs written independently of each other agree on every case, it is very unlikely that both are wrong in the same way. The second program also answers a question of its own: are the published rules, by themselves, enough to arrive at the agency's figures exactly?

We call the second program the reference program. It is written so that its arithmetic is exact. Computers normally store decimal numbers in a form that can be off by a tiny fraction, and in a calculation that rounds down at set points, a tiny fraction can change the answer by ten cents or a dollar. The reference program avoids this by holding every amount as a whole number of cents and every percentage as an exact fraction.

Both programs work from the same yearly figures. The reference program takes the average wage index, the taxable maximum and the cost-of-living increases from the tables inside the agency's source code. We did this on purpose. If the two programs ever disagreed, we wanted the disagreement to be about the method, not about one program having been given a different number. The bend points are the one exception. The reference program works them out from the average wage index by the rule in the law, and its results match the amounts the agency published for every year from 2021 to 2026.

Because both programs share these tables, a wrong number in a table would not show up as a disagreement between them. So we checked the tables separately, against the same figures as printed in a Congressional Research Service report (Congressional Research Service, 2026). The average wage index agrees in 73 of 73 years, the taxable maximum in 75 of 75, the bend points in 37 of 37, and the cost-of-living increases in 49 of 50. The one difference is the increase paid from January 2000, which lies outside the years this study uses.

3.3 The workers

The answer key covers 1,231 synthetic workers, in three groups. A synthetic worker is a test case that we wrote for this study. It is described by three things: a date of birth, the month in which the benefit is

claimed, and the amount earned in each year of work. These three things are all that a calculator needs to work out a benefit.

The first two groups have 450 workers each and were generated at random. We used the first group while writing the reference program and the checks, to find and fix our own mistakes. A program can end up fitted to the very cases it was developed on, and then pass them without being right in general. So, once the reference program and the checks were in their final form, we generated a second group of 450 in the same way, as a fresh test. The reference program agrees with the agency's calculator on all 450 of those as well (Section 3.4). The workers were born between 1959 and 1964 and claim anywhere from full retirement age to 60 months before it. Each group contains 50 workers of each of nine kinds of earnings history: steady earnings; rising earnings; earnings exactly at the taxable maximum; earnings above it; a history with gaps; a short history; a long one; a history with most of the earnings in the last years; and earnings given in dollars and cents. Birthdays were not spread evenly over the year. We made the first and second of a month, and 1 January, more common than chance would, because these are the birthdays where reaching an age on the day before the birthday changes the result.

The third group has 331 workers, and none of them is random. Each one was built deliberately to sit exactly where a rule changes, because that is where a program is most likely to slip. They cover five kinds of edge:

- birthdays: on the first or second of a month, on 1 January, and on 29 February;
- the formula: a monthly average just below a bend point, exactly at it, and just above it;
- the earnings limit: earnings exactly at the taxable maximum, one cent under it, and one dollar over it;
- the claim date: a claim exactly at full retirement age, and claims 35, 36, 37, 59 and 60 months early (the reduction rate changes after month 36, and 60 months is the earliest that a worker with a full retirement age of 67 can claim);
- the length of the earnings history: 12, 34, 35, 36 and 44 years (the calculation uses the best 35).

Taken together, the three groups contain 565 workers born on the first or second of a month, 71 born on 1 January and 14 born on 29 February, and 135 whose earnings include cents. Their monthly benefits run from $639 to $4,152. The highest of these equals the largest benefit that the agency says a worker can receive when claiming at full retirement age in 2026 (Social Security Administration, 2025), so the workers reach all the way to the top of the range.

3.4 Agreement

We ran all 1,231 workers through both programs and compared the three named amounts for each worker. The two programs return the same amount every time: for every worker, at all three stages (Table 2). The question raised in Section 3.2 therefore has a clear answer. The published rules are enough to arrive at the agency's figures exactly.

Table 2. Number of workers for whom the reference program and the agency's calculator return the same amount, at each of the three stages.

Group	Workers	Average indexed monthly earnings	Primary insurance amount	Monthly benefit
Random, first	450	450	450	450
Random, second	450	450	450	450
On rule edges	331	331	331	331
All	1,231	1,231	1,231	1,231

From here on, "the official result" means the figure that the agency's calculator returns for a worker. The reference program returns the same figure.

3.5 Can these workers tell a wrong program from a right one?

Two programs agreeing on every worker would mean little if the workers were too easy. A set of easy cases lets a faulty program pass as well as a correct one. So we tested the workers themselves. We made twelve copies of the reference program and put one deliberate mistake into each, of a kind a programmer could plausibly make. We then ran the twelve faulty copies on the first group of 450 workers and counted how many workers each mistake changed (Table 3).

Table 3. Twelve faulty copies of the reference program, each with one deliberate mistake, run on 450 workers.

The one mistake	Workers with a wrong amount at any of the three stages	Workers whose monthly benefit is wrong
Earnings adjusted to the year before eligibility, not two years before	389	389
Yearly increases before the claim ignored	389	389
Bend points of the claim year, not the eligibility year	388	388
Monthly benefit rounded to the nearest dollar	228	228
Increases compounded without rounding down each year	221	33
Primary insurance amount rounded to the nearest ten cents	219	28
Average rounded to the nearest dollar	213	50
One reduction rate for every early month	203	203
Highest 35 years chosen before the wage adjustment	103	102
Age counted from the birthday, not the day before	81	81
Average taken over the years worked, not 35	68	68
Taxable maximum not applied	50	50

The workers expose all twelve. Even the mistake that is hardest to see, leaving out the taxable maximum, changes the result for 50 of the 450 workers.

The table also shows why we compare calculators at three stages and not only at the final benefit. The three rounding mistakes in the middle rows each give a wrong amount at an earlier stage for more than 210 workers. Yet the monthly benefit is wrong for only 28 to 50 of them. The reason is the last step of the calculation. The benefit is rounded down to a whole dollar, and that rounding swallows most differences of ten cents. Someone who compared only the final benefit would see a small part of these mistakes.

This test is about the workers, not about any calculator. It shows that this set of workers, compared with the official results, exposes mistakes of all twelve kinds. Section 6 takes up a different question: which mistakes can be caught by someone who does not have the official results.

4. The open-source calculators

4.1 Which calculators, and how they were run

We looked for open-source programs that work out a retirement benefit from a worker's own earnings history. We searched the code-hosting sites GitHub and GitLab and four package registries (npm, PyPI, crates.io and r-universe). Seven projects were written independently of the agency's code. Five of them are still maintained, and we tested all five.

We left out three kinds of project: those no longer maintained; those derived from the agency's own source code, which should agree with the answer key and so tell us nothing; and those that ask the user to type in the primary insurance amount and do not compute it. The release lists every project we found, with the reason it was tested or left out.

The five we tested are listed below, each with the exact version tested, because these projects keep changing.

- **PolicyEngine-US** (version 2.145.5): a large model of United States taxes and benefits, written in Python. One part of it works out a Social Security benefit from an earnings record that the user supplies.
- **ssa.tools** (the code as it stood on 17 November 2026, commit ca05c52): a website that helps people plan when to claim, written in TypeScript.
- **social-security-calculator** (version 3.1.2): a small calculator that other programmers can build into their own software, written in JavaScript and distributed through the npm registry.
- **retiretui** (version 0.6.0): a retirement planner written in Rust.
- **The RetireGolden engine** (version 0.6.0): the calculation part of a retirement planner, also distributed through npm.

We treated every calculator in the same way. We read its source code first, installed it by itself, apart from the others, and changed nothing in it. For each calculator we wrote a short program that hands it the 1,231 workers one at a time, through the functions the calculator itself offers to outside users, and records what it returns. Four of the calculators give the average indexed monthly earnings and the primary insurance amount as well, so we compare them at all three stages. retiretui gives only the monthly benefit, so we compare it there.

To be fair to each project, we gave every calculator its input in the form that calculator expects. Five details matter.

- **ssa.tools** asks for “taxed earnings”. That is the figure on a worker’s Social Security statement, which never exceeds the yearly maximum. So we cut each year’s earnings to the maximum before handing them over.
- **retiretui** accepts earnings in whole dollars only. We rounded the 135 earnings histories that contain cents. The rounding changes none of their official monthly benefits, so the comparison is not affected.
- **The RetireGolden engine** has no single function that returns the monthly benefit. We took its primary insurance amount, applied the engine’s own reduction for an early claim, and rounded down to the dollar. No difference arises at that last step.
- **PolicyEngine-US** asks the user to state two things: that the worker has worked long enough to qualify, and that the rule which withholds benefits from people who are still working does not apply. We stated both, which matches the scope of this study.
- **social-security-calculator** applies the yearly increases up to the year before the current date on the computer. We ran it in 2026, when that is the right year for every worker in the set.

4.2 The result

Table 4 gives the result. One calculator, PolicyEngine-US, returns the official result for every worker. The other four differ for some workers. Counting the monthly benefit, which is the amount a person would be paid, ssa.tools differs for 26 workers (2 percent), social-security-calculator for 186 (15 percent), retiretui for 230 (19 percent) and the RetireGolden engine for 968 (79 percent). Counting a difference at any stage, the first three differ for about one worker in six.

Table 4. The five calculators compared with the agency’s calculator on 1,231 workers.

Calculator	Agrees at every stage compared (of 1,231)	Average differs	Primary insurance amount differs	Monthly benefit differs	What explains the differences
PolicyEngine-US	1,231	0	0	0	nothing to explain
social-security-calculator	1,042	168	171	186	does not apply the taxable maximum; births on 1 January; a slip in working out a date; a rounding slip
ssa.tools	1,023	2	208	26	one cause: rounds to the cent before rounding down
retiretui	1,001	not given	not given	230	works out the taxable maximum by formula; births on 1 January; rounds to the cent before rounding down
RetireGolden engine	259	972	972	968	two simplifications its source code states: counts only earnings from age 22 to 61; rounds each adjusted year down to the dollar

The three middle columns count the workers for whom each amount differs from the official one; a worker can be counted in more than one. The counts change from stage to stage because of rounding. In ssa.tools the primary insurance amount is ten cents off for 208 workers, and the final rounding down to a whole dollar leaves the monthly benefit wrong for only 26 of them. retiretui gives only the monthly benefit, so that is the one stage at which it can be compared.

PolicyEngine-US agrees at all three stages, including on the 331 workers placed on rule edges. That matters for reading the rest of the table. It shows that the workers are not so hard that no calculator can pass them: a calculator that follows the rules exactly gets every one of them right.

4.3 The causes

Rounding to the cent before rounding down (ssa.tools, retiretui). When an amount is raised by a yearly increase, the law rounds the result straight down to the next ten cents (Section 2). These two calculators do something slightly different. They first round the result to the nearest cent, which is what software that handles money usually does, and only then round down to ten cents. Most of the time the two ways give the same answer. They differ when the exact result falls just under a ten-cent mark, by less than half a cent. Here is a case from our workers. An amount of $2,210.60 is raised by 2.8 percent. The exact result is 2,272.4968. The law rounds that down to $2,272.40. A calculator that rounds to the cent first turns it into 2,272.50, and then there is nothing left to round down. Its result is ten cents too high.

In ssa.tools this is the only cause, and it explains all 208 of its differences. For 206 workers it happens at a yearly increase, as in the example above, and the primary insurance amount comes out ten cents too high. For the other 2 it happens one step earlier, where the total of the earnings is divided to give the monthly average: the same rounding to the cent lifts the average onto the next whole dollar. We confirmed this explanation directly. We took the official calculation, changed only the order of rounding, and it reproduced the ssa.tools figure for every one of the 208 workers. The effect on what a person would actually be paid is small. The monthly benefit is one dollar too high for 26 of these workers and unchanged for the other 182, because the final rounding down to a whole dollar hides most ten-cent differences.

retiretui rounds in the same order. Because retiretui gives only the final benefit, we could not see the ten-cent step directly, so we tested the explanation another way. We changed only the order of rounding in our reference program and compared its monthly benefit with retiretui's, on the 958 workers who are not touched by retiretui's other two causes. The two agree exactly for all 958. For 10 of those workers, the figure they share is one dollar above the official result.

Births on 1 January (social-security-calculator, retiretui). A person reaches an age on the day before the birthday, so the law treats someone born on 1 January as born in the previous year (Section 2). Both calculators take the year straight from the date of birth instead. social-security-calculator does so for the year of first eligibility, which sets the wage adjustment, the version of the formula and the yearly increases; its code for full retirement age gets the rule right. retiretui does so for the year of first eligibility and for the full retirement age as well. The introduction showed what this does to one worker.

In retiretui, all 71 workers born on 1 January get a wrong benefit, from $175 a month too low to $59 too high. In social-security-calculator, 66 workers are counted under this cause, from $130 a month too low to $59 too high; the other 5 born on that day also have earnings above the taxable maximum and are counted under that cause, below. Whether the benefit comes out too low or too high depends on the year of birth. The wage level, the formula and the yearly increases all move by a year, and they do not all push the benefit in the same direction.

The taxable maximum (social-security-calculator, retiretui). Only earnings up to the yearly taxable maximum count toward a benefit (Section 2). The two calculators go wrong here in different ways.

social-security-calculator does not apply the maximum at all. If a worker's earnings for a year are entered above the maximum, it counts them in full. Our set has 103 workers with such earnings, and 101 of them get a benefit that is too high. The typical difference (the median) is $597 a month, and the largest is $805. One point must be made in fairness. A year's earnings can be entered in two ways. Take someone who earned $200,000 in 2025, when the maximum was $176,100. That person can enter $200,000, the amount actually earned. Or they can enter $176,100, the figure printed on their Social Security statement, which never shows more than the maximum. With the second figure the package gives the right benefit. With the first it gives one that is too high, because it uses whatever number it is given. The package's documentation does not say which figure it expects, and our test entered the first. So this difference may be a missing warning in the documentation and not a mistake in the calculation.

retiretui does apply a maximum, but it does not use the figures the agency publishes. It works out each year's maximum itself, by taking the 1994 maximum and scaling it with the average wage index. For most years this gives the published figure. For 20 of the years from 1975 to 2026 it does not. The worked-out figure is too high in every year from 1975 to 1989, and in 2010, 2011 and 2016. Those last three are years in which the published maximum stayed where it was, because there had been no cost-of-living increase the December before; a formula based on wages alone does not know that. In 1990 and 1991 the worked-out figure is $300 too low. A worker whose earnings in one of these years fall between the two figures has some earnings counted that should not be, or some left out that should count. In our set this affects 149 workers who were not born on 1 January, and the effect on the monthly benefit is small: from $1 too low to $6 too high.

Two slips found only in social-security-calculator. The first is in working out the month in which a worker reaches full retirement age. The package starts from the day before the birthday and adds the right number of months. That works unless the starting day is the 29th, 30th or 31st and the month it lands in is shorter. The date 31 March plus one month, for example, should stay in April, but the programming language moves it on to 1 May. The package then places full retirement age a month later than it is, treats the claim as one month earlier than it is, and reduces the benefit for one month too many. Seventeen workers are affected, by $10 to $23 a month. In our set all seventeen were born on the first of a month, because the day before such a birthday is the last day of the month before. But the fault lies in the day of the month, not in the first: a worker born on 30 April 1959 shows it too (Table 5).

The second slip is in rounding down to ten cents. The package does this on an ordinary decimal number, which a computer stores only approximately. For three workers the exact amount is already a round

multiple of ten cents, such as 3,250.80. The computer holds it as a number a hair below, 3,250.7999..., and rounding down then gives 3,250.70. Ten cents are lost. For these three workers the monthly benefit happens to come out the same.

Two simplifications the authors state (RetireGolden engine). This engine departs from the law in two ways, and its source code states both, so they are choices and not oversights. First, it counts a worker's earnings only from the year the worker turns 22 to the year the worker turns 61. The law is wider. It counts earnings in any year after 1950 up to the year before the benefit starts. A worker who earned money before 22 or after 61 therefore has some earnings left out by the engine, and ends up with a lower average than the law gives. Second, it rounds each adjusted year of earnings down to the whole dollar, where the regulation rounds to the nearest cent, and that can lower the average by a dollar. We confirmed both by recomputation. The law with these two changes gives the engine's average for 1,230 of the 1,231 workers; the one worker left differs by a dollar and was not traced. All 972 of the engine's differences start at the average. The monthly benefit is too low for 968 of these workers. The typical shortfall (the median) is $24 a month, and the largest is $531. Among the 219 workers whose earnings all fall between 22 and 61, the engine agrees with the agency on 214, and the second simplification accounts for the other five.

4.4 Each cause on one simple worker

The 1,231 workers show how often each mistake occurs. To show what each mistake looks like, we also built eleven very simple workers. Each earns the same amount in every year from 1990 to 2025, so that only the one detail under test differs between a worker and its neighbour in the table. We ran them through the agency's calculator and through the four calculators shown in Table 5. The reference program agrees with the agency on all eleven.

Table 5. Monthly benefit in dollars for eleven simple workers, each earning the same amount in every year from 1990 to 2025. Figures that differ from the agency's are in bold. The RetireGolden engine is not shown, because its differences are the two simplifications its authors state.

Row	Worker	Agency	PolicyEngine-US	ssa.tools	social-security-calculator	retiretui
1	Born 15 June 1961, $32,000 a year, claims October 2026	2,019	2,019	**2,020**	2,019	**2,020**
2	Born 1 January 1960, $40,000 a year, claims January 2026	2,619	2,619	2,619	**2,538**	**2,508**
3	Born 2 January 1960, otherwise the same	2,493	2,493	2,493	2,493	2,493
4	Born 1 January 1964, $40,000 a year, claims January 2026	1,945	1,945	1,945	**1,983**	**1,983**
5	Born 2 January 1964, otherwise the same	1,971	1,971	1,971	1,971	1,971
6	Born 1 May 1959, $40,000 a year, claims January 2026	2,741	2,741	2,741	**2,726**	2,741
7	Born 30 April 1959, otherwise the same	2,741	2,741	2,741	**2,726**	2,741
8	Born 29 April 1959, otherwise the same	2,741	2,741	2,741	2,741	2,741
9	Born 1 March 1960, $40,000 a year, claims January 2026	2,478	2,478	2,478	**2,463**	2,478
10	Born 15 June 1962, $200,000 a year, claims October 2026	3,346	3,346	3,346	**5,480**	**3,348**
11	Born 15 June 1962, $40,000 a year, claims October 2026	2,231	2,231	2,231	2,231	2,231

The table reads as follows.

- **Row 1 is the rounding mistake.** ssa.tools and retiretui are one dollar too high.
- **Rows 2 to 5 are the 1 January birthday.** For a worker born on 1 January, social-security-calculator and retiretui give a wrong benefit: too low for the worker born in 1960, too high for the one born in 1964. For a worker born one day later, every calculator is right.
- **Rows 6 to 9 are the date slip in social-security-calculator.** It appears when the day before the birthday is the 29th, 30th or 31st of a month (births on 1 May, 30 April and 1 March) and not when it is the 28th (birth on 29 April).
- **Rows 10 and 11 are the taxable maximum.** With earnings of $200,000 a year, far above the maximum, social-security-calculator is $2,134 a month too high and retiretui is $2 too high. With $40,000 a year, every calculator is right.

In every case the mistake shows on one worker and is gone on the worker next to it, who differs in that one detail only.

4.5 What recurs, and what was reported

ssa.tools, social-security-calculator and retiretui were written by different people and, as far as their source code shows, share no code. Even so, two mistakes appear in more than one of them. Rounding to the cent before rounding down is in ssa.tools and in retiretui. Taking the year of birth straight from the date, for a worker born on 1 January, is in social-security-calculator and in retiretui. When unrelated programmers make the same mistake, the likely cause is not one person's carelessness. It is a rule that is easy to miss in the published method.

It also matters how far these mistakes reach. For an ordinary worker they hardly show at all. Our set has 498 workers who were born between the 3rd and the 28th of a month and never earned above a yearly maximum. For all 498, social-security-calculator returns the official monthly benefit. ssa.tools and retiretui each return it for 495 of them, and a benefit one dollar higher for the other three.

We reported these differences to the three projects. Each report gave the examples above and pointed to the lines of code concerned (ssa.tools issue 637, social-security-calculator issue 44 and retiretui issue 104; the addresses are in the reference list). We sent no report to the RetireGolden engine, because its difference is a choice its authors state.

5. Why the mistakes go unnoticed: published worked examples

Section 4 showed what the calculators get wrong. This section and the next ask why the mistakes were not noticed. A developer who writes a benefit calculator, and does not build the agency's program to test against, has no exact answer key. The natural substitute is a worked example: a published case that gives a worker's earnings, the steps of the calculation and the resulting amounts. A developer can run a new program on the same worker and see whether it produces the same amounts. We checked three sources of such examples against our answer key: a report written for Congress, the regulations themselves, and the agency's own website.

5.1 A Congressional Research Service report

The Congressional Research Service is the office that writes reports for members of Congress. One of its reports explains how a Social Security benefit is calculated (Congressional Research Service, 2026). The report follows five made-up workers, all born in 1960, who range from a very low earner to one who earned the taxable maximum every year. For each worker it prints the yearly earnings, the average indexed monthly earnings, the primary insurance amount, and the monthly benefit at each age at which the worker might claim. It then works through two what-if cases. In the first, each worker's highest year of earnings is taken away, as if the worker had spent that year out of work. In the second, the highest seven years are taken away.

We compared the amounts in the report's worked examples with what a correct program returns: for each of the five workers, the average, the primary insurance amount and the benefit at each claiming age, and the average and the primary insurance amount in each of the two what-if cases. We set aside the rows that depend on future yearly increases, which the report could only estimate and which therefore cannot be

checked. That leaves 55 printed amounts. A correct program reproduces 50 of them exactly. The other five all concern one worker, the maximum earner, and they come from two slips (Table 6).

The first slip is a rounding. The report gives the total of the maximum earner's 35 highest years of adjusted earnings as $4,801,018.60. Rounding each year to the cent first, as the regulation requires, gives $4,801,018.57; the three cents make no difference here. Divided by 420 months, either total gives 11,430.99 and a fraction. The law rounds the monthly average down to the whole dollar, which gives $11,430. The report's main table prints $11,431, which is the nearest dollar, and one of its appendix tables prints 11,430.90 for the same amount. The mistake then carries forward. The primary insurance amount worked out from that average is 20 cents too high, and the benefit at age 66 is one dollar too high.

The second slip is a skipped step. It is in the what-if case that takes away each worker's highest year. These workers have 41 years of earnings. Take one away and 40 are left, and the law still uses the best 35 of those. So the year that used to rank 36th now moves up into the 35 that count. For the maximum earner, the report takes the highest year out and puts nothing in its place: its total is exactly the original total minus that one year. That is the total of 34 years, not 35. The report's average comes out at $11,083, where the method gives $11,395, and its primary insurance amount at $3,305.60, where the method gives $3,352.40. For the other four workers in the same table, the report does bring in the 36th year, and its figures for them are right.

How do we know the slips are in the report and not in our own program? The agency's calculator settles it. On all five amounts it returns the same figure as our reference program, and not the figure in the report. The two slips are small. The other 50 amounts we checked are right to the cent, and the report's tables of yearly figures agree with the agency's in every entry but one, which lies outside the years we use (Section 3.2).

5.2 Examples in the regulations

The regulations that set out the method contain worked examples of their own. We checked two of them.

The first follows a worker called Ms. A, who reaches 62 in 1979 (20 CFR 404.211). The example lists her 28 years of earnings, adjusts them for wage growth, and arrives at average indexed monthly earnings of $903. Our reference program and the agency's calculator both return $903 for her, and both leave out the same five years of earnings that the example leaves out. This example is right.

The second example shows how a benefit is reduced when it is claimed early (20 CFR 404.410). A worker called Alex has a primary insurance amount of $980.50 and claims 44 months before full retirement age. The example works out the reduction as $228.80, subtracts it, and gives $751.70 as the monthly benefit. One step is missing. A monthly benefit that is not a whole dollar is rounded down to the dollar (42 U.S.C. 415(g); 20 CFR 404.304), so the benefit actually paid is $751. The arithmetic in the example is right as far as it goes. What it calls the benefit is the amount before the last rounding. A developer who tests against this example will find that a correct program, which returns $751, appears to be 70 cents out.

Table 6. The six published amounts that a correct program does not return, with the reason for each. “Report” is the Congressional Research Service report.

Where it is printed	Which amount	As printed	What the method gives	What went wrong
Report, maximum earner	Average indexed monthly earnings	$11,431	$11,430	Rounded to the nearest dollar, not down
Report, maximum earner	Primary insurance amount	$3,357.80	$3,357.60	Carried forward from the wrong average
Report, maximum earner	Benefit at 66	$3,704	$3,703	Carried forward from the wrong average
Report, maximum earner, highest year taken away	Average indexed monthly earnings	$11,083	$11,395	The 36th-highest year was not brought in
Report, maximum earner, highest year taken away	Primary insurance amount	$3,305.60	$3,352.40	Carried forward from the wrong average
Regulations (20 CFR 404.410), Alex	Monthly benefit	$751.70	$751	Stops before the last rounding down to the dollar

5.3 The agency’s own worked example

The agency’s website carries a worked example of its own, on two pages (Social Security Administration, 2026b). As it stood on 18 November 2026, it follows two workers who retire in 2027: one born in 1965 who claims at 62, and one born in 1960 who earned the taxable maximum every year and claims at full retirement age. The first page lists 40 years of earnings for each worker, from 1987 to 2026, with the adjustment factor for each year, the adjusted earnings, the total of the highest 35 years, and the average indexed monthly earnings. The second page applies the formula, adds the yearly increases and works out the monthly benefit.

We recomputed all of it by the method of Section 2. On the first page, all 80 adjustment factors and all 80 adjusted amounts agree with the page as printed, the same five years are left out for each worker, and the two averages agree: $6,072 and $11,843. On the second page, the bend points, both primary insurance amounts ($2,720.20 and $4,187.40) and both monthly benefits ($1,904 and $4,187) agree as well. The example is for a year later than the workers of this study, so the check takes two figures from the page itself that our tables do not yet hold: the average wage for 2025 and the yearly increase of December 2026.

The agency’s own example is right at every step. The slips in Table 6 belong to the other two sources.

5.4 What this means for a developer

Three things follow from these checks.

First, the slips in the published examples resemble the mistakes in the calculators. Two of the three are about rounding: one rounds to the nearest dollar where the law rounds down, and one stops before a last rounding. Rounding is also where three of the calculators in Section 4 go wrong. The people who write explanations of the method and the people who write programs from it stumble in the same places.

Second, a correct program does not pass these examples as printed. Picture a developer who tests a new program against the report. The program gives the maximum earner an average of $11,430, and the report says $11,431. One of the two is wrong, and the developer has to decide which. It is natural to trust the published report. A developer who does, and changes the program to round the average to the nearest dollar, now passes that example. The program is also now wrong for nearly half of all workers: in Table 3, that same change gives a wrong average for 213 of 450. Section 6.4 shows that this is more than a thought experiment: one of the calculators already returns the report's figure.

Third, the examples cover few kinds of worker. All five workers in the report were born in 1960, and none of them has earnings above the taxable maximum. No example gives a day of birth: the report and the agency's page give only a year, and the regulation examples give no date of birth at all. So none of the three sources contains a worker born on the first of a month or on 1 January. This matters because of what Section 4 found. The largest differences in the calculators come from two kinds of worker: one born on 1 January, and one with earnings above the taxable maximum. Neither appears in any published example. A developer could test a calculator against every one of these examples, pass them all, and still have both mistakes. Our set of workers contains both, which is how the mistakes came to light.

None of this makes worked examples a poor test. Section 6 shows that they catch more planted mistakes than any other check a developer can write. They are a strong check. They are not an answer key, and a developer should not treat them as one.

6. What a developer's own checks can and cannot see

Worked examples are one kind of test that a developer can use without the agency's program. There are other kinds, which a developer can write from the published rules alone. This section asks two questions about all of them. How many mistakes do such tests catch? And which mistakes get past them?

6.1 Four families of checks

A check is a small test that a program either passes or fails. We wrote four groups of checks, which we call families. Every one of them can be written from the published rules alone, without the agency's program (Table 7). Only one family needs to know the right answer in advance: the published examples, where it is printed in the report. The other three test a program without it.

Table 7. The four families of checks. All of them can be written from the published rules alone.

Family	What it checks	Does it need to know the right answer?
Basic checks	The program runs on 150 workers without failing; every amount is a whole dollar or a multiple of ten cents, as the law requires; no amount is negative; and no benefit is higher than the primary insurance amount it comes from	No
Published examples	The program returns the 50 amounts in the Congressional Research Service report that Section 5 found to be right	Yes: the amounts printed in the report
Relationships	Nine statements about how the results for two related workers must compare, for example that a worker who earns more never gets a lower benefit	No
Table checks	Six statements that the law makes about the tables of yearly figures, for example that the taxable maximum never falls	No

A relationship works like this. We run the same program twice, on two workers who differ in one known way, and check that the two results compare as the law says they must. No right answer is needed, only the comparison. The technique is known as metamorphic testing (Chen et al., 1998; Segura et al., 2016). The nine relationships are:

1. A worker who earns more never gets a lower result.
2. A worker who claims later never gets a lower benefit.
3. Earnings above the taxable maximum change nothing.
4. Adding a year with no earnings changes nothing.
5. Raising the earnings of one recent year by $4,200 adds exactly $10 to the monthly average ($4,200 divided by 420 months). It adds exactly $9.00, $3.20 or $1.50 to the primary insurance amount, depending on which slice of the formula the extra $10 falls in.
6. A claim 60 months early pays 70 percent of the primary insurance amount, and a claim 36 months early pays 80 percent, each rounded down to the dollar.
7. A claim at full retirement age pays the whole primary insurance amount, rounded down to the dollar.
8. A worker born on 1 January gets the same result as a worker born the day before, on 31 December.
9. A worker born on the first of a month reaches each age one month sooner than a worker born on the second.

The table checks test the tables of yearly figures that a calculator carries, not the calculation itself. Each one compares the figures with something the law says about them. The six are:

1. The bend points follow from the average wage index by a fixed rule.
2. The first bend point is lower than the second.
3. The taxable maximum is a multiple of $300 and never falls from one year to the next.
4. The three percentages of the formula are 90, 32 and 15.
5. Full retirement age rises by two months for each later year of birth, until it reaches 67.
6. No cost-of-living increase is negative.

One point about the published examples must be stated plainly. A developer without the answer key would test against all 55 amounts printed in the report, and a correct program fails five of them (Section 5). We use only the 50 that the answer key confirms. The figures below therefore show the published examples at their best, with the misleading amounts already taken out.

A check is only useful if a correct program passes it. A check that fails on a correct program would send a developer hunting for a mistake that is not there. So before using the checks, we ran every one of them on the reference program, which we know to be correct. In about 12,000 trials, no relationship was broken, no table check failed, and all 50 published amounts were reproduced.

6.2 Planting mistakes

To measure how good the checks are, we needed many faulty programs in which we knew exactly what was wrong. We made them automatically, a method known as mutation testing (DeMillo et al., 1978; Jia and Harman, 2011). A script took the reference program and produced 186 copies of it, each with one small change. The change might swap one arithmetic sign for another, raise or lower a number by one, turn a “greater than” into a “less than”, or replace one rounding step with another. The script also produced 121 copies of the tables of yearly figures, each with one entry left at the previous year’s value or altered slightly. That makes 307 changed copies, which we call variants. Each differs from the original in one place, and we know which.

We first ran each variant on the 1,231 workers and compared its results with the official ones. For 33 variants the change made no difference to any worker. We set those aside, because a change that never alters a result may not be a mistake at all. The other 274 variants give a wrong amount for at least one worker, so each of them is a real mistake. We then put those 274 through the four families of checks and recorded which checks each one failed (Table 8).

Table 8. How many of the 274 real mistakes each family of checks catches. One mistake can be caught by more than one family, so the rows do not add up.

	Changes to the program	Changes to the tables	All
Variants made	186	121	307
Of these, wrong for at least one worker (the real mistakes)	155	119	274
Caught by the published examples	130	91	221
Caught by a relationship	123	9	132
Caught by a basic check	64	1	65
Caught by a table check	0	34	34
Caught by at least one check that does not need the right answer (the three rows above, taken together)	126	34	160
Caught by at least one check of any kind	144	112	256
Caught by no check	11	7	18

Taken together, the checks catch 256 of the 274 mistakes and miss 18. The published examples do the most. They catch 221, and 96 of those are caught by nothing else. The checks that do not need the right

answer catch 160 without any help from the examples. They are strong on mistakes in the program: the relationships alone catch 123 of the 155. They are weak on mistakes in the tables, where the relationships catch 9 of 119. The reason is that a relationship compares a program with itself. A program that carries one wrong figure in a table still behaves consistently, only with the wrong number, so the comparison finds nothing to object to.

6.3 What gets through

Eighteen of the 274 mistakes pass every check. A developer who had written all four families of checks, and had no answer key, would have no sign that any of these 18 programs was wrong. Table 9 lists them, with the number of our workers that each one gets wrong.

Table 9. The 18 planted mistakes that pass every check.

The planted mistake	How many of the 18	Workers it gets wrong (of 1,231)
Each yearly increase rounded to the nearest ten cents, not down	1	866
Taxable maximum for 2022, 2023, 2024 or 2025 left at the previous year’s value	4	150 to 266
Average wage index for 1980 left at the 1979 value	1	157
Increase for December 2021 left at the previous year’s value, or off by a tenth of a percent	2	137
Small changes in how birthdays are handled at the turn of a year or on 29 February	7	14
Small changes in rounding each adjusted year of earnings to the cent	3	1 or 2

The first row is the most striking. A program that rounds each yearly increase to the nearest ten cents, where the law rounds down, gives a wrong primary insurance amount for 866 of the 1,231 workers. That is more than two workers in three, and none of our four families of checks notices it. There are two reasons. The relationships cannot see it, because the error is consistent: the two related workers that a relationship compares are both off in the same way. The published examples cannot see it either. The report prints the primary insurance amount only as it stands before any yearly increase, and it prints the later benefits in whole dollars. A ten-cent slip would show there only if it happened to push a benefit across a whole-dollar line, and for the workers in the report it does not.

The next three rows share one explanation. They are all figures in the tables that the published examples never use. The report’s five workers were all born in 1960, and its amounts rest on their earnings from age 21 to 61, which are the years 1981 to 2021. So the taxable maximum for a year after 2021, the average wage for a year before 1981, or the yearly increase of December 2021 plays no part in any amount the report prints. Such a figure can be wrong, and every published example will still come out right. The table checks do not help here either. A taxable maximum that was left at last year’s value is still a multiple of $300 and has not fallen, so it passes.

The last two rows are small by comparison. They change the result for at most 14 of the 1,231 workers, in rare cases such as a birthday on 29 February.

6.4 The checks applied to the calculators themselves

Planted mistakes are our own invention. To see whether the checks behave the same way on real programs, we applied them to the five calculators. The checks ask about 4,584 workers in all. We ran each calculator on those workers, through the same short programs as in Section 4, and applied every check to its answers.

A check was applied only where a calculator gives what the check needs. retiretui gives the monthly benefit alone, so the four relationships that compare the average or the primary insurance amount could not be applied to it. ssa.tools and social-security-calculator apply the yearly increases up to the current date, so the published benefits for claims in earlier years could not be compared for them. For ssa.tools, relationship 3 is satisfied by our own step of cutting earnings to the maximum before passing them in (Section 4.1). The table checks need a calculator's tables of yearly figures; we had the yearly maximum for two calculators and applied the one check that concerns it.

Table 10. The checks applied to the five calculators: which of the mistakes of Section 4 each family catches. The basic checks found nothing in any calculator.

Calculator	Mistake found in Section 4	Relationships	Published examples	Table check
PolicyEngine-US	none	none broken	all 50 confirmed amounts returned	not applied
social-security-calculator	does not apply the taxable maximum	**caught**: relationship 3 broken in 182 of 300 trials	not caught	not applied
social-security-calculator	births on 1 January	**caught**: relationship 8 broken in 300 of 300 trials	not caught	not applied
social-security-calculator	date slip; rounding slip	not caught	not caught	not applied
ssa.tools	rounds to the cent before rounding down	not caught	not caught; it returns three amounts the report prints wrongly	passes
retiretui	works out the taxable maximum by formula	**caught**: relationship 3 broken in 37 of 300 trials	**caught**: 2 of 24 benefits	**caught**: its maximum falls in 2011
retiretui	births on 1 January	relationship 8 could not be applied	not caught	not applied
retiretui	rounds to the cent before rounding down	not caught	not caught	not applied
RetireGolden engine	counts earnings from age 22 to 61 only	not caught	**caught**: 10 of 50 amounts	not applied

Four things stand out.

First, the calculator that matches the agency passes everything. PolicyEngine-US breaks no relationship in any trial, fails no basic check and returns all 50 confirmed published amounts. The checks do not raise false alarms on a correct program.

Second, the two simple relationships do their work on real code. Relationship 3, that earnings above the maximum change nothing, catches both treatments of the taxable maximum. Relationship 8, that a worker born on 1 January gets the result of a worker born the day before, fails in every one of 300 trials for social-security-calculator. These two relationships catch the two largest differences of Section 4.

Third, the rounding mistake passes everything on real code, as it did among the planted mistakes. ssa.tools breaks no relationship, fails no basic check, and returns every confirmed published amount that could be compared. It is the same kind of mistake as the first row of Table 9, met in practice.

Fourth, the published example does worse than miss that mistake. For the report's maximum earner, ssa.tools returns an average of $11,431, a primary insurance amount of $3,357.80 and a benefit at 66 of $3,704. These are exactly the three amounts that the report prints wrongly (Table 6). The reason is a coincidence of two slips: the report rounded 11,430.99 to the nearest dollar, and ssa.tools rounds it to the nearest cent first, which also gives 11,431. A developer who tests ssa.tools against the report sees agreement on the very amounts that a correct program fails.

The published examples catch two of the mistakes. They catch the age window of the RetireGolden engine, which no relationship catches, in the what-if case where the year at age 21 comes into the count. And they catch the worked-out maximum of retiretui, on the report's maximum earner.

For the remaining mistakes, the answer key is the practical way to find them. None of our four families tells rounding down apart from rounding to the nearest in the yearly increases. A developer could still catch that mistake with one hand-worked case that lands in the right place, such as $2,210.60 raised by 2.8 percent, but only by knowing in advance where to look. A comparison with the agency's figures needs no such knowledge. It does have to be made at the primary insurance amount, before the final rounding to a whole dollar hides the difference.

7. A pilot: a program written by an AI model

More and more programs of this kind are written with the help of an AI model, and some are written by a model that works from the text of the rules alone (Kennan et al., 2025). With an exact answer key, we can ask a question that is usually hard to answer: is a program written this way right? We ran a pilot to find out.

Setup. We used one model, Claude Sonnet 5.5. We gave it a package of official documents: excerpts of the law, seventeen sections of the regulations together with the definition of age, a table of the yearly figures, and a one-page sheet describing the task. Before any run, we checked that every one of the 24 rules that the reference program applies is stated somewhere in the package, so that the model was never asked for something the documents do not contain. Each run was a single request. The model had no tools and no access to the internet, and it was not shown the agency's calculator, our checks or any worked example. We fixed the plan, the method of scoring and the tools before the first run, so that nothing could be adjusted after seeing the results.

We ran the pilot in two parts, with five runs in each. In the first part, we had the model write the whole calculation as a program, as a developer would. In the second part, we did not ask for a program. We had

the model fill in the table of yearly figures that a program needs, 165 entries in all, from the documents. We then used each table with our own correct program, so that any wrong result could only come from a wrong figure in the table.

The programs. All five programs match the answer key. We scored each one on 781 workers: the second random group of 450, which had played no part in building our tools, and the 331 workers on rule edges. Every program returns the official result for every one of the 781 workers, at all three stages, and passes every check of Section 6. That includes the cases where the calculators of Section 4 go wrong: a birthday on 1 January, a birthday on the first of a month, and the order of rounding.

The tables of figures. Three of the five tables are right in all 165 entries. The other two each have one wrong entry, and it is the same entry in both: the full retirement age, in months, for workers born in 1959. The documents give that age as 66 years and 10 months, which is 802 months. One run wrote 82, with a digit dropped. The other wrote 796, which is the right figure for workers born in 1956. Both slips happened in turning years and months into months. With either table, the benefit is wrong for every worker whom the law treats as born in 1959: 67 of the 450 random workers and 11 of the 331 on rule edges. The checks caught both tables. Each fails the table check that full retirement age rises by two months for each later year of birth, and each breaks several relationships. The published examples would not have caught either one, because all of their workers were born in 1960.

What it shows. In this pilot, nothing faulty got past the checks. The five programs had no error that we could find, and the two faulty tables were caught. The pilot also shows where the risk lay. On the 781 workers, we found no mistake in the logic of any of the five programs. The only slips came in filling in a table, in a simple conversion of years and months into months. Two facts set the limits of what can be concluded. The pilot used one model and five runs in each part, which is enough to show that such errors occur and not how often. And the benefit calculation is widely documented, with public code available, so the model may have seen it before; leaving the worked examples out of a run does not change that.

8. What this means for someone building a benefit calculator

The findings point to a short list of practices for anyone who builds or maintains a benefit calculator. Each one is a familiar principle of careful programming. What this study adds is evidence of which ones matter for this calculation, and how much.

Test against the agency's own program. The agency offers its calculator as a ready-made program that can be downloaded and run, and it can take many cases at once. A developer who prefers not to set it up can use the workers released with this paper. Each comes with the agency's result at all three stages, so a new calculator can be compared with them directly.

Compare the amounts along the way, not only the final benefit. The last step rounds the benefit down to a whole dollar, and that hides most errors of ten cents. In Table 3, one rounding mistake gives a wrong primary insurance amount for 219 workers and a wrong monthly benefit for only 28. A test that looks at the benefit alone would miss it for the other 191.

Put test cases where the rules change. The cases that told the calculators apart were of three kinds: birthdays on 1 January, on the first of a month and on the 29th to the 31st; amounts that land just under a ten-cent or one-dollar mark; and earnings at or above the taxable maximum. Test workers should also come from more than one year of birth, because each year of birth uses different yearly figures.

Round the way the law says, and do it in whole numbers. A program that holds every amount as a whole number of cents, and rounds down in one step, avoids both rounding problems seen in Section 4. It does not round to the cent first, which can lift an amount by ten cents. And it does not rely on a decimal number that the computer stores a hair below its true value, which can lose ten cents.

Use the yearly figures as the agency publishes them. It is tempting to work a figure out by formula and save the trouble of updating a table. But the taxable maximum does not follow the average wage in every year. A figure worked out by formula will be right in most years and wrong in a few, and those few years matter to the workers who had high earnings in them.

Treat published examples as strong evidence, not as proof. When a carefully written program and a printed example disagree, the example may be the one that is wrong. Check the rule in the law before changing the program.

Write two simple checks. A worker born on 1 January must get the same result as a worker born the day before, on 31 December. And adding earnings above the taxable maximum must change nothing. Neither check needs to know the right answer, and each takes a few lines of code. Run on the calculators themselves, they caught the two largest differences found in Section 4 (Section 6.4).

Say what the calculator leaves out, and what it expects. The RetireGolden engine states in its source code that it counts earnings only from age 22 to 61. A user who reads that can judge whether it matters in their case. The same goes for what each input means. A calculator should say, for example, whether it expects earnings as they were earned or already cut to the taxable maximum.

For a person who simply uses one of these calculators, the findings are mostly reassuring. Take a worker born in the middle of a month whose earnings never went above the taxable maximum. For such a worker, three of the four calculators that differ from the agency returned the official benefit or one a dollar higher (Section 4.5). The larger differences arise only in particular cases: a birthday on 1 January, earnings above the taxable maximum, or years of work before age 22 or after 61. A user in one of those cases should check the result against the agency’s own calculator.

9. Limits

The calculators. We tested the five maintained, independently written, open-source calculators that our search found. Five further projects appeared in search listings and were not examined; the release names them. Commercial calculators, and others whose code is not public, are outside this study. The results describe the versions named in Section 4. The projects may have changed their code since, in part because of our reports.

The workers. The workers are synthetic, and we placed many of them deliberately where the rules change, because that is where mistakes show. A figure such as 1,023 of 1,231 therefore says how a

calculator does on this demanding set. It does not say what share of real claimants would see a wrong amount. This study does not measure that share. It is likely to be much lower, since most people are not born on the first of a month and do not earn above the taxable maximum.

The scope. The study covers retired workers born from 1959 to 1964 who claim in 2026, at or before full retirement age. It does not cover disability benefits, benefits for family members or survivors, the rule that withholds benefits from people who are still working, claims made after full retirement age, or the recalculation of a benefit in later years.

The answer key. The agency's Detailed Calculator is a public calculator, not the system that decides actual claims, though the agency reports that it matches the official calculations very well (Section 3.1). Our build matches the agency's ready-made program on every line, and a program written separately from the law matches both. All three use the same published tables of yearly figures, which we checked separately against a second source (Section 3.2).

How the causes were established. We confirmed the causes in two ways. Four causes are confirmed by recomputation: changing only that one thing in a correct program reproduces the calculator's figures. This holds for all 208 differences in ssa.tools, for the benefit of all 958 workers in retiretui who are outside its other two causes, for all 17 workers affected by the date slip in social-security-calculator, and for 1,230 of the 1,231 workers in the RetireGolden engine. The other causes are confirmed by reading the calculator's source code and by the simple workers of Table 5, where the mistake shows on one worker and is absent on its neighbour. These are the 1 January rule in both calculators and the two treatments of the taxable maximum. One worker in the RetireGolden engine, who differs by a dollar in the average, was not traced.

How the calculators were run. Section 4.1 describes how each calculator was given its input. Three choices are worth repeating here. retiretui could be compared only at the monthly benefit. The monthly benefit of the RetireGolden engine was put together from two of its functions. ssa.tools was given earnings already cut to the taxable maximum, which is what its input field asks for. Someone who called these calculators in a different way could get different results.

The checks. The four families of checks are our own design, and another developer would write different ones. The planted mistakes were all made by changing our reference program, so they are the kinds of mistake that can be made in a program written the way ours is. The 33 variants that changed no result for any worker were set aside as a group. We did not go through them to separate changes that are truly harmless from mistakes that our workers happen not to expose. On the calculators themselves, a check could be applied only where a calculator gives what the check needs: four relationships could not be applied to retiretui, published benefits for claims before 2026 could not be compared for two calculators, and only one table check could be applied, to two calculators.

The published examples. We examined one report, two examples in the regulations, and the agency's own worked example of two workers. Other worked examples exist that we did not examine. One is a page on the agency's website about workers who earned the taxable maximum.

The pilot. The pilot used one model and ten counted runs, on a calculation that the model may have seen before. It shows what happened in those runs, not how often such programs are right in general.

One author. One person read each calculator's source code, sorted the differences into causes and designed the checks. So that others can check each step, the release contains everything needed to do so: the workers, the official results, the programs that run each calculator, and the scripts behind every table.

10. Related work

Earlier studies of Social Security calculators. The closest earlier study comes from the agency itself. Martin and Kintzel (2016) reviewed six free online tools that help people decide when to claim: the agency's own Retirement Estimator, and tools from the Consumer Financial Protection Bureau, Boston College, AARP, Financial Engines and Bankrate. They described what each tool asks the user for, what it assumes and what it shows. They also entered two made-up workers into the five tools from outside the agency and compared the answers. The benefit estimates were roughly similar but not the same. For one worker, the estimates of the benefit at full retirement age ran from $2,240 to $2,750 a month. The authors traced the differences to what the tools assume, such as future pay rises and inflation, and to details such as one tool using an out-of-date earnings limit and another rounding full retirement age up to a whole year. The financial press reported the note in the same terms (Franklin, 2016), and one economist has argued that quick calculators leave out inputs that change the answer (Kotlikoff, 2015).

Our study differs from that one in two ways. The tools in that study start from a person's current salary and fill in the rest of the earnings history by assumption. We give each calculator a complete earnings history and a claim month, so nothing is left to assumption. And that study had no official figure to compare the tools with. We compare each result with the agency's own program, to the dollar.

The agency's published program has been used as a standard before, in one particular way. Several open-source projects are translations of that program into other programming languages. At least one group of them tests its translation against the original on thousands of generated cases and reports agreement to the cent (pyanypia, 2026). Such projects are copies of the answer key, which is why we left them out of Section 4. This paper uses the same standard for three things that, as far as we know, no earlier study has used it for: calculators written independently of the agency's code, published worked examples, and the checks a developer might write when the answer key is not at hand.

Testing without an answer key. A developer who writes a benefit calculator usually has no list of correct benefits to test it against, a difficulty known as the oracle problem (Barr et al., 2015). This paper uses three established ways of dealing with it.

The first is differential testing, which runs several programs on the same input and looks at where they disagree (McKeeman, 1998). Our comparison of the calculators is of this kind, with one difference: one of the programs, the agency's, can be trusted. The second is metamorphic testing, which checks how the outputs for related inputs compare (Chen et al., 1998; Segura et al., 2016). The nine relationships of Section 6 are tests of this kind. Tizpaz-Niari and colleagues applied the same technique to tax preparation software. They wrote their relations from tax publications with the help of tax experts, and found faults in an open-source tax program (Tizpaz-Niari et al., 2023). The third is mutation testing, which measures a set of tests by how many deliberately altered programs it detects (DeMillo et al., 1978; Jia and Harman, 2011). That is how Section 6 measures the checks.

Our contribution to this line of work is a measurement in one field. Because an exact answer key exists, we can say which planted mistakes the relationships miss. They turn out to be of the same kinds as the mistakes found in real calculators.

Law written as code. One line of research treats a law as something that can be written directly as a program a computer can run. Catala is a programming language designed for that purpose. When its authors used it to write out the French family benefits, they found a fault in the government's own implementation (Merigoux et al., 2021). OpenFisca and PolicyEngine are open-source models of tax and benefit rules built on the same idea. PolicyEngine-US is the calculator that matched the agency on every worker in this study. Escher and Banovic checked a state's online tool for screening benefit eligibility. They wrote the eligibility rules as a program themselves, generated test households, and compared the tool's answers with their own (Escher and Banovic, 2020). Our method is close to theirs. The difference lies in what serves as the truth. They had only their own reading of the rules to rely on. For Social Security, the agency's program exists, and our own program written from the law serves to confirm it.

AI models and benefit rules. Whether AI models can turn the text of a policy into working code is being actively studied. A team at Georgetown University ran experiments on the rules for food assistance and Medicaid. They concluded that the models help, but need outside knowledge and human review where the logic is complex (Kennan et al., 2025). Other work, on reasoning from statutes, has found that models make errors on questions that require applying the exact terms of a law (Blair-Stanek et al., 2023). Our pilot adds one thing to this work. It shows that, for a calculation that has an exact answer key, what a model produces can be checked against exact answers, worker by worker.

11. Conclusion

How accurate are open-source Social Security calculators? On the evidence of this study, they are close, and they are less exact than they appear. One of the five we tested returns the agency's result for every one of 1,231 workers, which shows that exact agreement can be achieved. Three others are right for ordinary workers and wrong in particular cases: a birthday on 1 January, an amount that falls a fraction of a cent short of a round figure, earnings above the yearly limit, and a year in which that limit did not move the way a formula predicts. The fifth leaves out part of a worker's earnings, and says so.

Most of the mistakes are small, and they are consistent in kind. They come from the fine details of the method, not from its main steps, and the same two appear in programs written by different people. The places a developer would turn to for reassurance have gaps of the same sort. One published worked example rounds the wrong way, and another stops before the last rounding. The checks we wrote from the rules alone catch the two largest mistakes in the real calculators and let the rounding mistake through. A developer who works from the published material can do everything reasonable and still be ten cents or a dollar out, or as much as $175 a month out for a client born on New Year's Day.

The remedy already exists. The agency publishes its calculator and the source code behind it. We have shown three things about it: that it can be built exactly as published, that a program written from the law alone agrees with it, and that it tells a calculator that is exactly right apart from one that is nearly right. The workers and official results released with this paper let anyone who maintains such a tool make the

same comparison without building anything. The same approach should work wherever a government agency publishes a program that carries out its own rules. How many agencies do so is a question worth asking.

Data availability

Everything needed to check or repeat this study is released with the paper (Thapa, 2026), at https://doi.org/10.5281/zenodo.24375957. The release contains:

- the three groups of synthetic workers and the eleven simple workers of Table 5, each with the agency's result at all three stages;
- the reference program and its tables of yearly figures;
- the script that builds the agency's calculator on Linux from its unchanged source code, and the comparison with the agency's ready-made program;
- the four families of checks, the scripts that made and tested the planted mistakes, and the answers of the five calculators to the workers the checks ask about;
- the short programs that run each of the five calculators, with the results each calculator returned;
- the list of every project found in the search, with the reason it was tested or left out;
- the scripts that check the published worked examples;
- the documents, task sheets, generated programs and scores of the pilot.

The agency's source code and the five calculators are not included. Each is identified by its version, its web address and a checksum, a short code that lets a reader confirm they have the same file.

Declarations

Funding. No funding was received for this work.

Competing interests. The author has no relevant financial or non-financial interests to disclose.

Ethics approval. Not applicable. The study uses synthetic data and public software, and involves no human participants.

Use of AI. AI-based tools assisted with the programming and the writing of this paper.

References

Barr, E. T., Harman, M., McMinn, P., Shahbaz, M., and Yoo, S. (2015). The oracle problem in software testing: A survey. IEEE Transactions on Software Engineering, 41(5), 507–525. doi:10.1109/TSE.2014.2372785.

Blair-Stanek, A., Holzenberger, N., and Van Durme, B. (2023). Can GPT-3 perform statutory reasoning? Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law (ICAIL 2023), 22–31. doi:10.1145/3594536.3595163.

Chen, T. Y., Cheung, S. C., and Yiu, S. M. (1998). Metamorphic testing: A new approach for generating next test cases. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of

Science and Technology. Reissued as arXiv:2002.12543.

Congressional Research Service (2026). Social Security: Benefit Calculation. Report R46658, updated 26 January 2026.

DeMillo, R. A., Lipton, R. J., and Sayward, F. G. (1978). Hints on test data selection: Help for the practicing programmer. Computer, 11(4), 34–41. doi:10.1109/C-M.1978.218136.

Escher, N., and Banovic, N. (2020). Exposing error in poverty management technology: A method for auditing government benefits screening tools. Proceedings of the ACM on Human-Computer Interaction, 4(CSCW1), Article 64. doi:10.1145/3392874.

Franklin, M. B. (2016). Social Security Administration evaluates online claiming tools. InvestmentNews, 13 December 2016.

Jia, Y., and Harman, M. (2011). An analysis and survey of the development of mutation testing. IEEE Transactions on Software Engineering, 37(5), 649–678. doi:10.1109/TSE.2010.62.

Kennan, A., Singh, L., Garcia Guevara, A., Ahmed, M., and Goodman, J. (2025). AI-Powered Rules as Code: Experiments with Public Benefits Policy. Beeck Center for Social Impact + Innovation (Digital Benefits Network) and Massive Data Institute, Georgetown University, March 2025.

Kotlikoff, L. (2015). The problem with quick and easy Social Security calculators. PBS NewsHour, 13 July 2015.

Martin, P. P., and Kintzel, D. (2016). A Comparison of Free Online Tools for Individuals Deciding When to Claim Social Security Benefits. Research and Statistics Note No. 2016-03, Social Security Administration, Office of Retirement and Disability Policy, December 2016. https://www.ssa.gov/policy/docs/rsnotes/rsn2016-03.html (read 18 November 2026).

McKeeman, W. M. (1998). Differential testing for software. Digital Technical Journal, 10(1), 100–107.

Merigoux, D., Chataing, N., and Protzenko, J. (2021). Catala: A programming language for the law. Proceedings of the ACM on Programming Languages, 5(ICFP), Article 77. doi:10.1145/3473582.

pyanypia (2026). A translation of the Social Security Administration's Detailed Calculator into Python. https://github.com/anthonycolavito/pyanypia (read 17 November 2026).

Segura, S., Fraser, G., Sanchez, A. B., and Ruiz-Cortés, A. (2016). A survey on metamorphic testing. IEEE Transactions on Software Engineering, 42(9), 805–824. doi:10.1109/TSE.2016.2532875.

Shoven, J. B., and Slavov, S. N. (2014). Does it pay to delay Social Security? Journal of Pension Economics and Finance, 13(2), 121–144. doi:10.1017/S1474747213000309.

Social Security Administration (2025). 2026 Social Security Changes (fact sheet). https://www.ssa.gov/news/en/cola/factsheets/2026.html

Social Security Administration (2026a). Social Security Detailed Calculator, version 2026.2, released 13 July 2026, with source code. Office of the Chief Actuary. https://www.ssa.gov/oact/anypia/ (description at https://www.ssa.gov/oact/anypia/description.html, read 18 November 2026).

Social Security Administration (2026b). Benefit Calculation Examples for Workers Retiring in 2027. Office of the Chief Actuary. https://www.ssa.gov/oact/progdata/retirebenefit1.html and retirebenefit2.html (pages dated 14 October 2026, read 18 November 2026).

Thapa, S. J. (2026). Replication package for "How Accurate Are Open-Source Social Security Calculators?" Zenodo. doi:10.5281/zenodo.24375957.

Tizpaz-Niari, S., Monjezi, V., Wagner, M., Darian, S., Reed, K., and Trivedi, A. (2023). Metamorphic testing and debugging of tax preparation software. IEEE/ACM 45th International Conference on Software Engineering:

Software Engineering in Society (ICSE-SEIS), 138–149. doi:10.1109/ICSE-SEIS58686.2023.00019. arXiv:2205.04998.

Statute and regulations: 42 U.S.C. 402(a), 402(q), 415(a), (b), (e), (g), (i), 416(l), 430; 20 CFR 404.102, 404.211, 404.304, 404.410.

Software tested: PolicyEngine-US 2.145.5; ssa.tools (commit ca05c52); social-security-calculator 3.1.2; retiretui 0.6.0 (commit 9723643); @retiregolden/engine 0.6.0.

Reports filed with the projects, 17 November 2026: ssa.tools issue 637, https://github.com/Gregable/social-security-tools/issues/637; social-security-calculator issue 44, https://github.com/cmcnulty/SocialSecurityCalculator/issues/44; retiretui issue 104, https://github.com/kenianbei/retiretui/issues/104.