



Owl - Optimal wealth lab: Mathematical Foundations of a Retirement Planning Optimizer

*A mixed-integer linear programming approach to
US federal and state tax-aware retirement wealth management*

2026 Edition

Martin-D. Lacasse

August 11, 2026

© 2024–2026 Martin-D. Lacasse

Contents

1	Introduction	1
2	US Retirement Tax Rules and Their Implementation in Owl	3
2.1	Overview	3
2.2	Savings Accounts	3
2.3	Household Financial Profile and Cash Flow	5
2.4	Social Security Benefits	7
2.5	Pension Income	9
2.6	Single Premium Immediate Annuity (SPIA)	10
2.7	Roth Account Rules	11
2.8	Health Savings Accounts	13
2.9	ACA Marketplace Premiums (Pre-65)	15
2.10	Medicare and IRMAA Premiums	17
2.11	Long-Term Capital Gains Tax	18
2.12	Net Investment Income Tax	19
2.13	Ordinary Income Tax	20
2.14	State Income Tax	21
3	Model Architecture and Solution Approach	23
3.1	Overview	23
3.2	Planning Horizon and Time Steps	23
3.3	Account Dynamics	24
3.4	Asset Allocation and Glide Paths	25
3.5	Spending Profile and Objectives	25
3.6	The Self-Consistent Iteration Loop	26
3.7	Filing Status and the Death of a Spouse	28
3.8	Solvers	28
3.9	MIP Decomposition	28

4	Indices, variables, and parameters	30
4.1	Indices	30
4.2	Variables	31
4.3	Parameters	35
4.4	Intermediate variables	44
5	Formulation with imposed asset allocation ratios	50
5.1	Constraints	51
5.1.1	Required minimum distributions (RMDs)	51
5.1.2	Income tax brackets	51
5.1.3	Standard exemption	51
5.1.4	Withdrawal limits	52
5.1.5	Withdrawal ordering	52
5.1.6	Posthumous account activities	53
5.1.7	Roth conversions	53
5.1.8	Minimum taxable balance	53
5.1.9	Initial balances	54
5.1.10	Roth 5-year maturation and conversion ladder	54
5.1.11	Cash-flow surplus	54
5.1.12	Account balances	55
5.1.13	Net spending	59
5.1.14	LTCG bracket constraints	60
5.1.15	Net investment income tax	61
5.1.16	Taxable ordinary income	63
5.1.17	State income tax	63
5.1.18	Social Security taxability	64
5.1.19	Medicare brackets and costs	66
5.1.20	ACA marketplace premium constraints	69
6	Mapping of decision variables	71
6.1	Reverse mapping of indices	75
7	Building constraint matrices	76
7.1	Inequality constraints	76
7.2	Equality constraints	79
7.3	Other considerations	83
7.4	Constraint tags and the explanation engine	86

8	Objective Functions and Their Formulation	90
8.1	Maximizing net spending	90
8.2	Maximizing bequest	92
8.3	Lexicographic tie-breaking	92
9	Rate models for return scenarios	93
9.1	Historical data	93
9.2	Constant rate methods	94
9.3	Deterministic varying rates: <code>historical</code>	94
9.4	Stochastic models	95
9.4.1	Inflation skewness correction	95
9.4.2	Mean anchoring (<code>constrain_mean</code>)	96
9.4.3	Multivariate normal: <code>historical_gaussian</code> and <code>gaussian</code>	97
9.4.4	Log-normal models: <code>lognormal</code> and <code>historical_lognormal</code>	97
9.4.5	Bootstrap sequence-of-returns: <code>historical_bootstrap</code>	99
9.4.6	Vector autoregression: <code>vector_ar</code>	100
9.4.7	DCC-GARCH(1,1): <code>garch_dcc</code>	101
9.4.8	Gaussian mixture model: <code>gmm</code>	102
9.4.9	Hidden Markov model: <code>hmm</code>	103
9.4.10	Gaussian copula: <code>historical_copula</code>	105
9.5	User-supplied rates: <code>dataframe</code>	107
9.6	Monte Carlo and stress testing	107
10	Stochastic Spending Optimization and Longevity Risk	109
10.1	Scenario bases	109
10.2	The commitment LP and efficient frontier	110
10.3	Longevity risk	111
10.4	Scenario generation	112
10.5	Implementation in Owl	112
11	Optimization of Social Security Claiming Ages	114
11.1	The claiming-age grid and benefit table	114
11.2	Claiming-age adjustment factors	115
11.3	Binary claiming variables and LP constraints	115
11.4	Spousal and survivor benefits in the SC loop	116
11.5	Implementation in Owl	117
A	Source File Reference	119

Abstract

This document describes the mathematical foundations of **Owl** - Optimal wealth lab, a Python application that optimizes US retirement planning using mixed-integer linear programming. Owl models federal tax laws, Medicare and ACA marketplace premiums, Social Security rules, pension income, Roth conversions, Health Savings Accounts, and account dynamics jointly within a single optimization framework. The self-consistent iteration loop and MIP reformulations handle nonlinear tax interactions. Chapters 1–3 present the conceptual model for a general audience; Chapters 4–9 provide the formal mathematical formulation, constraint structure, and rate models for implementers and extenders.

1. Introduction

This document describes the mathematical model underlying the optimization algorithms implemented in Owl, which is a Python application optimizing retirement planning using mixed-integer linear programming. Owl is designed for US retirees as it considers US federal tax laws, Medicare premiums, rules for 401k including required minimum distributions, maturation rules for Roth accounts and conversions, Social Security rules, etc. The goal of these calculations is to optimize the financial aspects of retirement planning, considering the types of savings accounts, federal income tax, contributions, return rates, Medicare premiums, Roth conversions, and desired income among many other things.

The approach is described here mathematically and the Python implementation follows the structure and notation presented in this document. The intent of this document is to provide a guide to the source code for any individual desiring to extend the model to other cases. The mathematical description is not tied to any specific mathematical programming language so that it remains as generic as possible and can be solved using different linear solvers. The source code, installation instructions, and usage examples are available on the project’s GitHub repository [1].

This document is intended for two audiences. Chapters 1 through 3 require only basic financial literacy: familiarity with retirement account types, tax-deferred vs. tax-free savings, Social Security, and Medicare is sufficient to follow the discussion. Chapters 4 through 9 require mathematical maturity: comfort with linear algebra, index notation, and the basics of linear and mixed-integer programming is assumed. Chapter 9 additionally assumes familiarity with probability distributions and time-series models. Readers interested only in the conceptual model may stop after Chapter 3; readers seeking to extend or verify the implementation will need the later chapters.

The remainder of this document is organized as follows. Chapter 2 surveys the US retirement tax rules that Owl models—Social Security benefits, pension income, Roth account rules, ACA marketplace premiums, Medicare premiums, capital gains tax, the Net Investment Income Tax, and ordinary income tax—with notes on how

each is implemented and where known limitations apply. Chapter 3 describes the overall model architecture: how account balances evolve over time, how asset allocation glide paths are specified, how spending objectives are formulated, and how the self-consistent iteration loop resolves the nonlinear tax interactions that arise within a linear programming framework. Chapters 4 through 7 present the formal mathematical formulation: the indices, variables, and parameters; the constraint structure; the mapping of decision variables into the solver’s one-dimensional array; and the construction of the constraint matrices. Chapter 8 defines the objective functions. Chapter 9 describes the rate models available for generating annual return sequences, from simple fixed-rate assumptions to fully stochastic simulations. Appendix A provides a reference guide to the Python source files.

2. US Retirement Tax Rules and Their Implementation in Owl

2.1 Overview

Retirement income in the United States comes from four broad account types: taxable brokerage accounts, tax-deferred accounts such as traditional 401(k)s and IRAs, tax-free Roth accounts, and Health Savings Accounts (HSAs). Withdrawals from these accounts interact with multiple simultaneously computed taxes and premiums: ordinary income tax (with bracket stacking), long-term capital gains (LTCG) tax, the Net Investment Income Tax (NIIT), ACA Premium Tax Credit, Medicare Part B premium surcharges, and the taxability fraction of Social Security (SS) benefits. Because all these quantities depend on the same underlying income variables, optimizing any one strategy in isolation—such as Roth conversion timing or Social Security claiming age—can inadvertently worsen another.

Owl models all of these rules *jointly* within a single linear program (LP) or mixed-integer program (MIP) so that strategies such as Roth conversions, SS claiming age, and withdrawal order are optimized together rather than sequentially. Many rules are nonlinear (piecewise, min/max); Owl handles them within a linear framework using a self-consistent (SC) iteration loop for quantities that can be lagged by one iteration, and MIP reformulations for quantities that require exact simultaneous computation.

2.2 Savings Accounts

US retirement savings are held in four fundamentally different account types, each with its own tax treatment, contribution rules, and withdrawal constraints. Owl models all four types and optimizes across them jointly. For married couples, each spouse holds their own independent set of accounts; balances, contribution histories, claiming ages, and Required Minimum Distribution (RMD) schedules are tracked

separately for each individual while the optimizer considers the combined household portfolio.

Taxable brokerage accounts

Taxable accounts hold securities purchased with after-tax dollars. Dividends and interest are taxed as ordinary income (or at preferential qualified-dividend rates) in the year received, regardless of whether the cash is withdrawn. Capital gains are recognized and taxed only when securities are sold. There is no tax deduction for contributions, no contribution limit from a federal tax standpoint, and no RMD requirement.

Owl models the taxable account as generating a dividend and interest yield each year proportional to the balance. Capital gains arise when the optimizer elects to sell holdings; all realized gains are treated as long-term (see the limitation noted in Section 2.11). When an initial cost basis is provided via `setCostBasis()`, the unrealized gain fraction is computed from the actual basis and tracked throughout the plan via a pro-rata update rule (see Eqs. (4.26)–(4.28)); this reduces the taxable gain recognized on withdrawals in the early plan years.

Tax-deferred accounts (traditional IRA, 401(k), and 403(b))

Tax-deferred accounts accept pre-tax contributions that reduce current taxable income, subject to annual IRS limits. Growth inside the account is not taxed until withdrawal. Every dollar withdrawn is taxed as ordinary income in the year taken. Required Minimum Distributions (RMDs) apply starting at age 70–75 depending on the account owner’s birth year (see Section 2.13). A 10% early withdrawal penalty applies to withdrawals taken before age 59½, subject to IRS exceptions not modeled here.

Owl supports contributions to traditional 401(k) and IRA accounts as entered in the Wages and Contributions table described in Section 2.3. The optimizer determines the optimal annual withdrawal amount from these accounts subject to RMD floors and other constraints. The pro-rata rule (IRS Form 8606) is not modeled: Owl treats the entire tax-deferred balance as pre-tax, so every dollar converted or withdrawn is fully taxable (see the limitation in Section 2.7).

Tax-free Roth accounts (Roth IRA and Roth 401(k))

Roth accounts accept after-tax contributions. Qualified withdrawals of both principal and earnings are entirely tax- and penalty-free. Roth accounts are not subject to RMDs during the original owner’s lifetime. Roth conversions—transfers from a tax-deferred account to a Roth account—are fully taxable as ordinary income in the year of conversion but produce tax-free growth thereafter. The detailed five-year seasoning and age-59½ rules governing Roth withdrawals and conversions are described in Section 2.7.

Owl supports contributions to Roth 401(k) and Roth IRA accounts, and optimizes Roth conversion amounts annually subject to user-specified constraints and tax-bracket considerations.

Health Savings Accounts (HSA)

HSAs are the only triple tax-advantaged savings vehicle in the US tax code: contributions are pre-tax (reducing ordinary income and MAGI), growth is tax-free, and qualified medical withdrawals are entirely tax-free at any age. After age 65 (Medicare enrollment), non-qualified withdrawals are taxed as ordinary income, effectively making the HSA a second traditional IRA without Required Minimum Distributions. A surviving spouse inherits the account intact, preserving its full tax-advantaged status. Medicare Parts B and D premiums, IRMAA surcharges, Medigap premiums, and out-of-pocket medical costs all qualify as HSA withdrawals, so these expenses can be funded entirely tax-free from the account. Annual contribution limits are set by the IRS (approximately \$4,400 for self-only and \$8,750 for family coverage in 2026) and must cease at Medicare enrollment. See Section 2.8 for the detailed rules and implementation.

2.3 Household Financial Profile and Cash Flow

Owl takes two primary inputs. The first is the **case configuration file** (a `.toml` file), which specifies all plan parameters: account balances, Social Security claiming ages, spending objectives, rate assumptions, solver options, and so on. The second is the **Household Financial Profile (HFP)**, an Excel workbook that captures the year-by-year financial data for each individual and the household as a whole. The HFP consists of three tables: the *Wages and Contributions* table (one tab per individual), the *Fixed Assets* table, and the *Debts* table.

Wages and Contributions table

Each individual has their own sheet with the following year-indexed columns:

- *year* — calendar year;
- *anticipated wages* — earned income expected for that year, entered net of all amounts listed in the *ctrb* columns (employer portions included in the *ctrb* columns are not subtracted, as they were never part of wages). This single amount serves both as cash available to the plan and as taxable employment income; as a known simplification, income tax on the dollars contributed to Roth and taxable accounts is therefore not modeled during the working years;
- *other inc* — other ordinary income not captured elsewhere (enter 0 if not applicable);
- *net inv* — rent and trust distributions that count as net investment income for NIIT (enter 0 if not applicable);
- *taxable ctrb* — contributions to the taxable brokerage account;
- *401k ctrb* — traditional 401(k) contributions;
- *Roth 401k ctrb* — Roth 401(k) contributions;
- *IRA ctrb* — traditional IRA contributions;
- *Roth IRA ctrb* — Roth IRA contributions;
- *HSA ctrb* — Health Savings Account contributions (enter 0 if not applicable; automatically zeroed at Medicare enrollment age; entries past age 65 are ignored);
- *Roth conv* — planned or historical Roth conversion amounts; the five years preceding the plan start date are used to implement the five-year lookback rule (Section 2.7);
- *big-ticket items* — large planned expenses such as a home purchase or college tuition that must be funded from the portfolio in a specific year.

Fixed Assets table

The Fixed Assets table describes assets held outside investment accounts: real estate (including a primary residence), fixed annuities, collectibles, precious metals, or individual stocks. Each row specifies whether the asset is active, its name, type, acquisition year, cost basis, current value, expected return rate, year of disposal, and sales commission. The optimizer accounts for sale proceeds and any resulting capital gains as fixed assets are disposed of during the plan horizon.

Debts table

The Debts table lists outstanding loan and mortgage obligations. Each row specifies whether the debt is active, its name, type (loan or mortgage), start year, term, original principal, and interest rate. Annual debt-service payments enter the cash-flow constraint as fixed outflows.

Cash-flow balance

In every plan year, Owl enforces an annual cash-flow balance. Spending plus all taxes, Medicare premiums, and debt payments must equal income from wages, Social Security benefits, account withdrawals, fixed-asset sale proceeds, and other income. Account contributions do not appear in this balance: anticipated wages are entered net of contributions (see the Wages and Contributions table above), and the contributed amounts are deposited directly into their accounts through the account-evolution equations. Big-ticket expenses shift additional spending into specific years. The optimizer determines the combination of account withdrawals and Roth conversions—across all account types and both spouses—that satisfies this constraint while maximizing the chosen objective.

2.4 Social Security Benefits

Rules

Monthly Social Security retirement benefits equal the Primary Insurance Amount (PIA) multiplied by an age-based adjustment factor. The Full Retirement Age (FRA) depends on birth year: 65 for those born before 1938; 66 for those born between 1943 and 1954; and 67 for those born in 1960 or later, with two-month-per-year interpolation for the transition cohorts (born 1938–1942 and 1955–1959). A special Social Security Administration (SSA) rule treats individuals born on January 1st as attaining age on the last day of the prior month, shifting their effective FRA by one month. Reference: SSA POMS RS 00615.003 [2].

Benefits claimed before FRA are permanently reduced: by 5/9 of 1% per month for the first 36 months before FRA, and by 5/12 of 1% per month for any additional months beyond that, yielding a minimum of approximately 70% of PIA at age 62 when FRA is 67. Benefits deferred beyond FRA earn delayed retirement credits of 8% per year up to age 70. Reference: SSA POMS RS 00615.015 [2].

The spousal benefit is up to 50% of the higher-earning spouse’s PIA, reduced by the claiming spouse’s own PIA (floored at zero), with an additional reduction of 25/36 of 1% per month for claiming before the spousal FRA (which mirrors the own FRA schedule) and no bonus for deferral beyond FRA.

The survivor benefit is the greater of the deceased spouse’s actual benefit and 82.5% of the deceased’s PIA, reduced if the survivor claims before their own survivor FRA (minimum 71.5% at age 60, interpolated linearly in the claiming month). Unlike the retirement benefit, it earns no delayed retirement credits past the survivor FRA, so deferring beyond that date is never advantageous. The survivor benefit and the survivor’s own retirement benefit are distinct entitlements with independent claiming dates, and only the larger of the two is paid in any given month; claiming one does not forfeit the other. A survivor may therefore claim the survivor benefit at 60 and let their own benefit accrue delayed credits to age 70, or the reverse. Reference: CFR § 404.338 [2].

The fraction of SS benefits subject to federal income tax ranges from 0% to 85% based on “provisional income” (PI), defined as adjusted gross income (excluding Social Security) plus tax-exempt interest plus half of total SS benefits. Thresholds are \$25,000/\$34,000 (single) and \$32,000/\$44,000 (MFJ), frozen since 1983 and 1994 respectively. Reference: IRS Publication 915 [3].

Owl also accepts a `trim_pct` and `trim_year` to model a projected reduction in SS benefits, for example to reflect a potential trust fund depletion scenario.

Implementation in Owl

SS benefit amounts, FRA schedules, spousal benefits, and survivor benefits are computed from the PIA inputs and each individual’s date of birth. Own, spousal, and survivor entitlements are built as three separate time series and then combined as SSA pays them: while both spouses are alive, each receives their own benefit plus any excess spousal amount; from the year of the first death n_d the spousal add-on ends and the survivor receives $\max(\zeta_{isn}^{\text{own}}, \zeta_n^{\text{surv}})$. The survivor claiming date is set by the parameter `survivor_claim_age`, which accepts "immediate" (as soon as eligible), "FRA", or an explicit age; it is never earlier than age 60 or n_d , and is capped at the survivor FRA. When SS claiming ages are optimized (`withSSAges="optimize"`) and the first-to-die’s claiming age is not itself a decision variable, ζ_n^{surv} is a constant and is folded directly into the own-benefit table $B_{ikn}^{\text{own}} \leftarrow \max(B_{ikn}^{\text{own}}, \zeta_n^{\text{surv}})$ for $n \geq n_d$, making the post-death payout exact for every candidate claiming age k . Otherwise the survivor amount depends on the deceased’s chosen k and is carried as the non-negative excess-survivor offset $\max(0, \zeta_n^{\text{surv}} - B_{iskn}^{\text{own}})$ updated by the SC loop. SS

taxability is computed iteratively using the provisional income formula, updated at each pass until it converges; a more exact method is also available via the option `withSSTaxability="optimize"` [3].

Limitations and assumptions

The SSA family maximum benefit cap (CFR § 404.403) is not modeled; this may overstate benefits when multiple beneficiaries claim on one worker's record. The SSA earnings test is not modeled; users are assumed to be fully retired before their FRA. The survivor claiming date is a user setting rather than an optimized decision: the model evaluates whichever date is specified, but does not search for the one that maximizes the objective. A worker who dies before claiming is credited with the 82.5% PIA floor rather than the benefit accrued to the date of death. The Windfall Elimination Provision (WEP) and Government Pension Offset (GPO) were repealed by the Social Security Fairness Act (signed January 5, 2025) and therefore require no modeling.

2.5 Pension Income

Overview

A pension is a fixed periodic payment from an employer-sponsored defined-benefit plan, paid for the life of the retiree. Unlike Social Security, pension amounts are set by the plan sponsor and are not subject to federal phase-in rules; they begin in full at the elected commencement age. Some plans offer cost-of-living adjustments (COLAs) that partially or fully index payments to inflation; many do not.

Implementation in Owl

Each individual may have a monthly pension amount and a commencement age. Owl converts the monthly amount to an annual figure and determines the first plan year in which payments begin, using the individual's birth year and birth month to compute the age reached in each calendar year. In the first year of receipt, the annual amount is prorated by the fraction of the year remaining after commencement. For subsequent years the full annual amount (twelve times the monthly benefit) is credited.

When indexed, the annual pension payment grows with inflation each year so that its purchasing power remains constant in real terms. When not indexed, the

nominal dollar amount is fixed for the entire duration of the pension. In both cases no pension income is credited before the commencement year. Pension income enters the cash-flow constraint as ordinary income alongside wages, other income, and Social Security benefits.

Joint-and-survivor option

When a pension holder elects a joint-and-survivor (J&S) option, the surviving spouse receives a fraction (e.g., 50%, 75%, or 100%) of the primary's pension after the primary's death. The user specifies the *actual* monthly amount received (already actuarially reduced if J&S was elected) and the survivor fraction as a number between 0 and 1. Starting in the year after the first spouse passes away, the surviving spouse receives that fraction of the deceased's pension each year for the remainder of their horizon. If the primary's pension is inflation-indexed, the survivor benefit follows the same cost-of-living adjustment, so both the primary's pension (while alive) and the survivor benefit grow at the same rate over time. The resulting survivor benefit is added to the surviving spouse's pension income and enters the cash-flow as ordinary income.

Limitations and assumptions

Early-commencement reductions and late-commencement credits are not modeled.

2.6 Single Premium Immediate Annuity (SPIA)

Overview

A Single Premium Immediate Annuity (SPIA) converts a one-time lump-sum premium into a guaranteed income stream that begins immediately and continues for the annuitant's lifetime. Unlike a pension, the income rate (premium-to-income ratio) is market-determined and must be supplied by the user; Owl does not contain an actuarial pricing model. The premium is funded from the individual's tax-deferred account via an IRA rollover—a non-taxable transfer—so no ordinary income is recognized at the time of purchase. Annual payments begin in the year of purchase and are fully taxable as ordinary income for the remainder of the plan horizon.

Implementation in Owl

Each individual may hold one or more SPIAs; each is added independently via `addSPIA()`. In the purchase year, the premium is deducted from the annuitant's tax-deferred balance as a non-taxable IRA rollover; no ordinary income is generated at that time. If the purchase year precedes the plan's start year (an already-purchased SPIA), the premium deduction is skipped and income begins at plan year zero.

Annual SPIA income equals twelve times the specified monthly benefit. When the *indexed* flag is set, the nominal benefit grows with the inflation series each year so that its real purchasing power remains constant; otherwise the nominal dollar amount is fixed for the life of the annuity.

An optional *survivor fraction* ($0 \leq f_s \leq 1$) directs a specified share of the annuitant's benefit to continue to the other individual after the annuitant's death.

SPIA income enters the cash-flow constraint as ordinary income alongside wages, pension payments, and Social Security benefits. It therefore raises Modified Adjusted Gross Income (MAGI) and affects Medicare IRMAA surcharges and the Social Security taxability fraction.

Limitations and assumptions

- The premium is assumed drawn entirely from a tax-deferred account (IRA rollover); funding from taxable or Roth accounts is not modeled.
- Full annual income is credited from the purchase year; partial-year proration in the purchase year is not modeled.
- Annuity pricing (premium-to-income ratio) is user-supplied; Owl contains no actuarial pricing model.
- Payments cease at the annuitant's plan horizon; term-certain and period-certain riders are not modeled.

2.7 Roth Account Rules

Rules

Roth IRA contributions are made with after-tax dollars; qualified withdrawals are entirely tax- and penalty-free. Reference: IRC § 408A [4]. Contribution *principal* may be withdrawn at any time without tax or penalty. Roth *earnings* on contributions require both (a) the account must be at least five years old and (b) the

owner must be at least age $59\frac{1}{2}$ (or meet another IRS exception) for a penalty-free withdrawal.

Each Roth *conversion* carries its own independent five-year clock. Conversion principal withdrawn before age $59\frac{1}{2}$ is subject to a 10% early withdrawal penalty if taken within five years of that conversion. Roth conversions are fully taxable as ordinary income in the year of conversion. Reference: IRC § 408A(d)(3); IRS Publication 590-B [5].

Roth IRA accounts are not subject to Required Minimum Distributions (RMDs) during the original owner's lifetime. Roth 401(k) RMDs were eliminated by SECURE 2.0 Act § 325 effective for tax years beginning after December 31, 2023. A 10% early withdrawal penalty also applies to tax-deferred account withdrawals before age $59\frac{1}{2}$ (subject to exceptions not modeled here).

Implementation in Owl

The five-year maturation rule is enforced as a set of minimum-balance constraints that retain recent conversions at compounded value, plus gains-only on recent contributions, as a floor on the Roth account balance. The age- $59\frac{1}{2}$ threshold is computed per individual from their birth month. Conversion amounts are added to taxable ordinary income in the year of conversion. Conversion behavior can be further controlled via the options `maxRothConversion`, `noRothConversions`, `startRothConversions`, and `swapRothConverters` [4, 5].

Limitations and assumptions

A single unified five-year lookback is applied conservatively: recent conversions at compounded value plus gains-only on contributions are retained as a minimum balance floor. Contribution principal is not separately tracked, so some valid early withdrawals of contribution principal are conservatively disallowed. The pro-rata rule (IRS Form 8606) is not modeled: Owl treats the entire tax-deferred balance as pre-tax, so every dollar converted or withdrawn is fully taxable ordinary income. Users with significant IRA after-tax (nondeductible) basis should be aware of this limitation. Roth conversions are also disallowed in the last two years of the plan horizon.

2.8 Health Savings Accounts

Rules

A Health Savings Account (HSA) is a tax-advantaged account available to individuals enrolled in a High-Deductible Health Plan (HDHP). HSAs offer a unique triple tax advantage: contributions are pre-tax (above the line, reducing AGI), investment growth is tax-free, and withdrawals used for qualified medical expenses are tax-free at any age. After age 65, non-qualified withdrawals are taxed as ordinary income—making the HSA functionally equivalent to a traditional IRA for general use, but without Required Minimum Distributions. Reference: IRC § 223 [6]; IRS Publication 969 [7].

Eligibility requires enrollment in a High-Deductible Health Plan (HDHP) and no disqualifying coverage (e.g., Medicare, non-HDHP insurance). Enrollment in any part of Medicare (Part A, B, C, or D) immediately disqualifies one from contributing; existing HSA balances may still be withdrawn for qualified expenses. Annual contribution limits for 2026 are \$4,400 for self-only coverage and \$8,750 for family coverage, with an additional \$1,000 catch-up for individuals age 55 and older (each spouse with family coverage may add \$1,000 to their own account if both are 55+). Contributions must cease once the account owner enrolls in Medicare, which typically occurs at age 65. Reference: IRC § 223(b)(7) [6]; IRS Publication 969 [7].

A surviving spouse who is the named beneficiary inherits the HSA intact and may treat it as their own HSA, preserving all three tax advantages. A non-spouse beneficiary must include the full account balance in taxable income in the year of inheritance. Reference: IRC § 223(f)(8)(B) [6].

Implementation in Owl

Owl models the HSA as a fourth savings account type alongside taxable, tax-deferred, and Roth accounts. Because the dominant retiree use is funding qualified medical expenses, all HSA withdrawals are treated as tax-free (simplified model); see Limitations below. The key modeling choices are as follows.

- *Contributions are pre-tax.* As with all contributions, anticipated wages are entered net of HSA contributions (Section 2.3), which excludes the contributed dollars from ordinary taxable income, from provisional income used to compute Social Security taxability, and from the modified adjusted gross income (MAGI) used in the two-year Medicare IRMAA lookback. Contributions are zeroed automatically at Medicare enrollment age; HFP entries past age 65 are ignored.

- *Withdrawals are tax-free and fund any expense.* HSA withdrawals enter the annual cash-flow balance at full dollar value, identical to Roth account withdrawals, and are excluded from taxable income. Because Medicare Parts B and D premiums, IRMAA surcharges, Medigap premiums, and other out-of-pocket medical costs are all qualified expenses under IRS Publication 969, the user may apply HSA withdrawals to cover these costs tax-free. The optimizer does not track medical expenses explicitly; it treats HSA withdrawals as generic tax-free cash flow, and the user is responsible for ensuring actual withdrawals correspond to qualified expenses.
- *No Required Minimum Distributions.* Unlike tax-deferred accounts, HSAs are never subject to mandatory annual distributions.
- *Cannot exceed account balance.* As with taxable and Roth accounts, the optimizer may not withdraw more than the current HSA balance in any year.
- *Bequest at heirs' tax rate.* The HSA balance remaining at the end of the plan is valued after the heirs' assumed marginal income tax rate, identical to the treatment of tax-deferred accounts. A surviving spouse inherits the HSA intact with no tax (the full balance transfers to their own HSA), but a non-spouse beneficiary must include the full amount in ordinary income. The reported estate value applies the same discount.
- *Asset allocation.* In account-specific allocation mode, the HSA defaults to the same allocation as the Roth account, reflecting that both grow tax-free. Users may also specify a separate HSA allocation.
- *Spousal transfer at death.* At the death of the first spouse, the full HSA balance transfers to the surviving spouse's HSA account, preserving its tax-advantaged status.

Limitations and assumptions

Withdrawals. All HSA withdrawals are treated as qualified medical expenses and therefore entirely tax-free. Non-medical withdrawals after age 65—which would be taxable as ordinary income—are not separately tracked. Before age 65, non-qualified withdrawals incur ordinary income tax plus a 20% penalty; Owl does not model this penalty because all withdrawals are assumed qualified. This simplification is conservative for the dominant retiree use case: funding Medicare premiums and out-of-pocket health costs. Future work may introduce a more detailed model that distinguishes medical from non-medical withdrawals.

Contributions. The annual IRS contribution limit (\$4,400 self-only and \$8,750 family for 2026) is not enforced as an optimizer constraint; users are responsible

for not exceeding it in the *HSA ctrb* column of the Household Financial Profile. The catch-up contribution (\$1,000 for age 55+) is not automatically applied; users aged 55+ must add it manually if desired. Self-only vs. family coverage is not distinguished; the applicable limit is the user’s responsibility. The mandatory contribution stop at Medicare enrollment is enforced by zeroing contributions from that year forward. `setHSA()` is invoked either explicitly or automatically when loading from configuration (e.g., TOML). HFP entries in the *HSA ctrb* column for years when the individual is age 65 or older are overwritten with zero. If Medicare enrollment occurs mid-year, the IRS allows prorated contributions for months before enrollment [8]; Owl zeroes the entire year, which is conservative.

Eligibility. Owl assumes the user is HSA-eligible (enrolled in an HDHP with no disqualifying coverage). HDHP status, disqualifying coverage, and dependent status are not verified.

Bequests. Non-spouse beneficiary inheritance is modeled by applying the heirs’ marginal income tax rate to the terminal HSA balance, identical to the treatment of tax-deferred accounts. The simplified qualified-withdrawal model (all HSA withdrawals tax-free during the plan horizon) is not extended to non-qualified withdrawals after age 65; those remain deferred to a future path.

2.9 ACA Marketplace Premiums (Pre-65)

Rules

Before Medicare eligibility at age 65, individuals and couples may obtain health coverage through the Health Insurance Marketplace (Exchange) established by the Affordable Care Act. The Premium Tax Credit (PTC) reduces the net premium cost for Marketplace plans based on household modified adjusted gross income (MAGI) and the Federal Poverty Level (FPL). Reference: IRC § 36B [9].

The applicable percentage—the share of income a household is expected to contribute toward the benchmark plan—varies with the ratio of MAGI to FPL. The IRS publishes the applicable percentage table annually. For 2025, Rev. Proc. 2024-35 provides the table under the Inflation Reduction Act (IRA), which suspends indexing and caps the contribution at 8.5% for income above 400% FPL. Reference: Rev. Proc. 2024-35 [10].

For 2026 and beyond, Rev. Proc. 2025-25 provides the applicable percentage table when the IRA enhanced subsidies have expired: indexing resumes, contribution percentages are higher, and there is no subsidy for income above 400% FPL (the “subsidy cliff” returns). Reference: Rev. Proc. 2025-25 [11].

Federal Poverty Level guidelines are published annually by the Department of Health and Human Services (HHS). For 2026, the FPL for the 48 contiguous states and DC is \$15,960 (single) and \$21,640 (couple). Reference: HHS Poverty Guidelines [12].

Household size for ACA purposes is the number of individuals in the tax household who are under age 65 and require coverage (Medicare-eligible individuals are excluded). Below 138% of FPL, individuals in Medicaid expansion states generally qualify for Medicaid rather than Marketplace subsidies.

Implementation in Owl

Owl models ACA net premium costs for pre-65 years when the user supplies the annual benchmark Silver plan (SLCSP) premium via `setACA()`. Two computation modes are available.

In the default `withACA="loop"` mode, ACA costs are computed after each solver pass using `acaCosts()`, which applies the IRS applicable-percentage table to MAGI and FPL for each plan year. Rules are year-aware: plan years before 2026 use the 2025 table; 2026 and later use the 2026 table. Below 138% FPL, Owl returns the full SLCSP (no PTC), reflecting Medicaid eligibility in expansion states.

In `withACA="optimize"` mode, ACA bracket selection and cost are folded into the optimization. The formulation uses 7 MAGI brackets (6 intervals up to 400% FPL, plus one above) with SOS1 binary selection, MAGI decomposition, and a piecewise cost: brackets 0–5 use proportional contribution rates; bracket 6 (above 400% FPL) imposes the full SLCSP. ACA uses current-year MAGI (no two-year lag like Medicare IRMAA). Unlike IRMAA and the NIIT, the ACA MAGI is the full-Social-Security variant $\mathbb{G}_n^{\text{aca}}$ (IRC §36B adds the non-taxable portion of Social Security back to AGI).

Limitations and assumptions

The SLCSP benchmark premium is user-supplied and inflated by the plan’s inflation factor. FPL and applicable-percentage tables are updated annually in the code; manual updates are required when HHS or IRS publish new values. If Congress extends IRA subsidies (e.g., the 8.5% cap above 400%), parameters must be updated accordingly. Below 138% FPL, Owl assumes Medicaid-eligible and returns full SLCSP; expansion-state rules only.

2.10 Medicare and IRMAA Premiums

Rules

Medicare Part B eligibility begins at age 65. The standard 2026 monthly premium is \$202.90 (\$2,434.80/year). Reference: Centers for Medicare & Medicaid Services (CMS) 2026 Medicare Part B premium announcement [13].

The Income-Related Monthly Adjustment Amount (IRMAA) adds a surcharge based on MAGI from two years prior. In 2026, the five incremental per-person per-month surcharges are \$81.20, \$121.70, \$121.70, \$121.70, and \$40.70, applied at MAGI thresholds of \$109k, \$137k, \$171k, \$205k, and \$500k (single); these thresholds are doubled for married filing jointly (MFJ). Reference: CMS 2026 IRMAA tables [13].

MAGI for IRMAA purposes is the beneficiary's adjusted gross income (AGI, IRS Form 1040 line 11) plus tax-exempt interest income (line 2a). It therefore includes all ordinary taxable income, LTCG, and dividends, but only the *taxable* portion of Social Security benefits (the up-to-85% already in AGI); the non-taxable remainder is *not* added back. This differs from the Affordable Care Act definition (Section 2.9), which does add back non-taxable Social Security. Reference: SSA POMS HI 01101.010 [2]. For couples, premiums reflect both individuals when both are Medicare-eligible. Filing status transitions from MFJ to single at the first spouse's death.

Implementation in Owl

Owl determines which IRMAA bracket applies to each year's MAGI and computes the exact cumulative premium for that bracket. Two computation modes are available. The default, `withMedicare="loop"`, computes IRMAA costs after each solver pass. Setting `withMedicare="optimize"` folds bracket selection directly into the optimization for an exact result. The two-year MAGI lookback is fully implemented. Because IRMAA in plan year n depends on MAGI from year $n - 2$, the first two plan years require MAGI values from before the plan start. These are supplied by the user via the `previousMAGIs` option as a list of the two MAGI values from two and one years before the plan start. When not supplied, both default to zero, which may underestimate IRMAA surcharges in the first two plan years for higher-income households. Reference: [13].

Part D and assumptions

Medicare Part D is modeled using the same MAGI brackets and two-year lookback as Part B. Part D IRMAA surcharges (CMS 2026) are included by default; the optional

Part D base premium is configurable (default zero). Users can disable Part D via `includeMedicarePartD=false` when they have other drug coverage (e.g., employer or VA). IRMAA thresholds in Owl are assumed to scale with the general inflation rate; in practice, CMS adjusts brackets annually but not necessarily by the Consumer Price Index (CPI).

2.11 Long-Term Capital Gains Tax

Rules

Long-term capital gains (LTCG) and qualified dividends are taxed at preferential federal rates of 0%, 15%, or 20% based on total taxable income (ordinary income plus LTCG combined). Reference: IRS Topic 409 [14]; IRS Publication 550. LTCG “stacks on top of” ordinary income: the portion of gains that pushes combined taxable income above each threshold is taxed at the corresponding higher rate. For 2026, the 0%/15% threshold is \$49,450 (single) or \$98,900 (MFJ); the 15%/20% threshold is \$545,500 (single) or \$613,700 (MFJ). These thresholds are indexed for inflation annually.

Implementation in Owl

LTCG bracket allocation is always active (no option required). Total gains are split across the three rate brackets; because the rates increase (0%, 15%, 20%), the optimizer naturally uses the lowest-rate brackets first. The upper limit of each bracket depends on how much ordinary income already occupies the corresponding threshold; these limits are refined at each solver pass until they converge. The total LTCG tax is derived from the bracket allocation after solving. Reference: [14].

An optional exact MIP formulation is available via `withLTCG="optimize"`. In this mode, the ordinary taxable income G_n is embedded as a continuous LP variable together with binary variables that select the applicable LTCG bracket in each year, replacing the self-consistent loop refinement of bracket thresholds with an exact mixed-integer constraint.

Limitations and assumptions

Short-term capital gains (assets held one year or less) are taxed as ordinary income. Owl does not separately model short-term versus long-term gains within the taxable account; all realized gains are treated as long-term.

2.12 Net Investment Income Tax

Rules

The Net Investment Income Tax (NIIT) is a 3.8% surtax imposed by the Affordable Care Act on net investment income (NII) for taxpayers whose MAGI exceeds \$200,000 (single) or \$250,000 (MFJ). Reference: IRC § 1411 [15]; IRS Form 8960. The thresholds are *not* indexed for inflation. The tax equals 3.8% of the lesser of net investment income and the amount by which MAGI exceeds the applicable threshold. NII includes dividends, interest, LTCG, rents, and royalties, but excludes wages, Social Security benefits, tax-deferred retirement distributions, and Roth withdrawals.

Implementation in Owl

The NIIT is computed annually from MAGI and net investment income (dividends, interest, LTCG, and any rent or trust income entered in the `netinv` column). Rent and trust income from `netinv` is included in NII and also flows into cash-flow and taxable-income as ordinary income. The tax uses the IRS formula described above. Because the NIIT depends on the solution, it is held fixed during each solver pass and updated from the new income values until it converges. The threshold is fixed in nominal dollars; its real impact over time is correctly reflected in the model. Reference: [15].

An optional exact MIP formulation is available via `withNIIT="optimize"`. In this mode, MAGI is embedded as a continuous LP variable together with a binary variable z_n^j that selects whether the NIIT threshold is exceeded each year. The full IRS $\min(\cdot, \cdot)$ cap—capping the NIIT base at net investment income when ordinary income drives MAGI well above the threshold—is modeled exactly without a second binary via a continuous surplus variable $\delta_n^j \geq 0$; because the optimizer minimizes taxes, it naturally drives δ_n^j to its upper bound, reproducing the correct IRS value.

Limitations and assumptions

None.

2.13 Ordinary Income Tax

Rules

The US federal income tax is a progressive tax applied to taxable ordinary income (adjusted gross income, or AGI, minus deductions). Reference: IRS Publication 17 [16]. Under the One Big Beautiful Bill Act (OBBBA, signed July 4, 2025), seven marginal brackets apply at rates of 10%, 12%, 22%, 24%, 32%, 35%, and 37%. Standard deductions for 2026 under the OBBBA are \$16,100 (single) and \$32,200 (MFJ). An additional deduction of \$2,050 (single) or \$1,650 (MFJ) per eligible individual applies at age 65 or older.

OBBBA § 1002 provides an extra \$6,000 per-person senior bonus deduction for taxpayers aged 65 or older, phasing out at \$6 per \$100 of MAGI above \$75,000 (single) or \$150,000 (MFJ) and fully phased out at \$175k/\$250k. This bonus expires after tax year 2028. Reference: [17].

Required Minimum Distributions (RMDs) are mandatory annual withdrawals from tax-deferred accounts, beginning at age 70 for those born before 1949, age 72 for those born 1949–1950, age 73 for those born 1951–1959, and age 75 for those born in 1960 or later (effective 2033), per SECURE 2.0 Act § 107 [18]. RMD fractions are taken from the IRS Uniform Lifetime Table III (effective 2022), as published in IRS Publication 590-B [5], except when the sole designated beneficiary is a spouse more than ten years younger, in which case IRS Table II (Joint and Last Survivor) applies. Roth accounts are exempt from RMDs. Filing status transitions from MFJ to single in the calendar year of the first spouse’s death.

Owl can also model a future reversion to pre-OBBBA tax rates (10%, 15%, 25%, 28%, 33%, 35%, 39.6%) from a user-specified calendar year (`yOBBBA` parameter), enabling scenario analysis of potential future tax law changes.

Implementation in Owl

Ordinary income is allocated across the seven tax brackets, with bracket widths and rates updated each year for inflation and for the OBBBA sunset schedule. RMD fractions are computed per individual and year from the IRS Uniform Lifetime Table. The OBBBA 65+ bonus deduction phaseout is computed from the current MAGI and refined at each solver pass. Reference: [16, 18, 17].

Limitations and assumptions

State income taxes are modeled separately; see the next section. Itemized deductions are not modeled; only the standard deduction (plus the 65+ and OBBBA bonus deductions) is applied. The OBBBA 65+ bonus deduction phaseout is computed outside the LP (as a post-solution update) because it is a nonlinear function of MAGI. The IRS Uniform Lifetime Table III is used for standard cases. When the spouse is the sole designated beneficiary and more than ten years younger than the account owner, IRS Table II (Joint and Last Survivor, Pub. 590-B) is applied instead. Inherited IRA and beneficiary RMD rules are not modeled.

2.14 State Income Tax

Rules

Most US states levy a personal income tax on top of the federal tax, each with its own bracket structure, standard deduction, and filing statuses. The treatment of retirement income varies widely: the majority of states exempt Social Security benefits, several exempt all retirement income, and many others offer capped exclusions for pension or retirement-account income. A few states exempt Social Security only below an income threshold.

Implementation in Owl

A plan may specify a two-letter state code. Bracket rates and widths, standard deductions, Social Security treatment, and retirement-income and pension exemption caps for all 50 states and the District of Columbia are read from a data table (`data/taxes_state.toml`) with separate entries for Single and Married Filing Jointly filers; the filing status switches to Single at the passing of a spouse, and bracket widths and deductions are indexed to inflation through γ_n . State taxable income is derived from quantities already present in the model: federal ordinary taxable income, plus capital gains (which most states tax as ordinary income), minus the exempted portion of Social Security, pension exemptions, the state standard deduction, and any capped retirement-income exemption. The resulting state tax joins the cash-flow constraint alongside its federal counterpart. The LP formulation is given in Sec. (5.1.17). States without an income tax simply generate no state-tax variables.

Limitations and assumptions

State AGI is approximated from federal quantities; state-specific additions and subtractions beyond those listed above are not modeled. Capital gains are taxed as ordinary income, which is accurate for most but not all states. States that exempt Social Security only below an AGI threshold (e.g., Kansas, Missouri) are treated with a fixed exempt-or-taxed status rather than the threshold rule. Local and county income taxes, state tax credits, and itemized deductions are not modeled.

3. Model Architecture and Solution Approach

3.1 Overview

Owl formulates retirement planning as a linear program (LP) or, when binary variables are required, a mixed-integer program (MIP). The plan is divided into annual time steps covering the entire household planning horizon. In each year, the optimizer simultaneously determines withdrawal amounts from all accounts, Roth conversion amounts, and spending levels, subject to tax laws, account balance dynamics, and user-specified constraints. The resulting solution determines the optimal sequence of financial decisions across the full horizon in a single solve—not year by year.

A complication arises because several tax quantities depend nonlinearly on income, yet also influence the cash-flow balance that income must satisfy. Owl resolves this circularity through a self-consistent (SC) iteration loop: nonlinear quantities are fixed at their previous-iteration values during each LP solve, then recomputed from the new solution, and the process repeats until convergence. For quantities that require exact simultaneous treatment, binary variables and MIP reformulations replace the SC loop entirely.

3.2 Planning Horizon and Time Steps

The plan horizon runs from the current calendar year (year 0) through the year following the last individual’s projected death. Each year index corresponds to one calendar year; all decision variables are resolved at annual granularity.

For a single individual, the horizon spans the years from the plan start through the assumed year of death, which is derived from a user-supplied life expectancy or target age. For married couples, the shorter-lived spouse’s death marks the transition from a joint plan to a survivor plan. Both death years are inputs to Owl; they are

treated as certain within a given scenario. Monte Carlo and historical stress-test simulations run the optimizer repeatedly across many return scenarios. When the longevity-risk option is enabled in stochastic spending optimization (Chapter 10), ages at death are drawn from SSA period life tables for each scenario, so that the planning horizon itself becomes a random variable.

3.3 Account Dynamics

Each individual maintains up to four savings accounts: a taxable brokerage account, a tax-deferred account, a Roth account, and a Health Savings Account (HSA). A married couple therefore has up to eight accounts in total, tracked and optimized jointly.

At the start of each plan year each account balance grows by the year’s investment return for that account’s asset allocation. During the year the following flows are applied:

- contributions from the Wages and Contributions table are deposited (modeled at mid-year so they earn half a year of return);
- withdrawals are made to fund spending, pay taxes, and cover other expenses including debt service and big-ticket items;
- Roth conversions transfer a chosen amount from the tax-deferred account to the Roth account, adding the converted amount to that year’s ordinary taxable income;
- the taxable account generates a dividend and interest yield proportional to its balance, which is taxed as ordinary income or at preferential qualified-dividend rates;
- Required Minimum Distributions enforce a minimum withdrawal from each tax-deferred account once the owner reaches the applicable RMD starting age;
- any spending surplus—income exceeding expenses in a given year—is deposited back into the taxable account;
- HSA contributions are pre-tax (netted out of anticipated wages) and cease at Medicare enrollment; HSA withdrawals fund qualified medical expenses entirely tax-free and are not included in taxable income;
- at the death of the first spouse, a user-specified fraction of each account is transferred to the surviving spouse’s corresponding account, reflecting beneficiary designations; the surviving spouse inherits the HSA intact at full tax-advantaged status.

The balance at the start of the following year is the result of all of these flows. All accounts are assumed to be rebalanced to their target asset allocation at the end

of each year.

3.4 Asset Allocation and Glide Paths

The target allocation of each account across asset classes (equities, corporate bonds, Treasury notes, and cash) is specified by the user. Allocation can be prescribed at three levels of granularity: per individual per account, per individual across all accounts combined, or identically for the entire household.

When the user specifies starting and ending allocation values, Owl interpolates smoothly over the plan horizon using either a linear glide path or an S-curve (hyperbolic tangent) glide path. The S-curve produces a gradual, natural transition that accelerates through the midpoint and tapers at both ends, which is often a more realistic model of how investors de-risk their portfolios over time. For couples, each individual’s allocation glides independently over their own planning horizon; the survivor’s allocation therefore continues from whatever value it had reached at the moment of the first spouse’s death.

3.5 Spending Profile and Objectives

Spending profile

Net spending in each year follows a user-specified *spending profile* that scales a common first-year basis. The profile encodes the desired shape of spending over time; the optimizer determines the overall scale—the basis—that is achievable given the household’s resources and constraints.

Common profiles include a flat profile (constant real spending throughout retirement), a *smile* profile (higher spending in early and late retirement, lower in the middle years), and a linearly decreasing profile. At the death of the first spouse, spending is multiplied by a user-specified survivorship factor to reflect the reduction in household expenses for a single person.

Primary objectives

Two optimization objectives are supported:

Maximize net spending.

Subject to a desired bequest—which may be zero—the optimizer maximizes the sum of inflation-adjusted annual spending over the full plan horizon. This is

the most common objective for retirees who want to sustain the highest possible standard of living.

Maximize bequest.

Subject to a fixed annual spending level, the optimizer maximizes the after-tax estate value delivered to heirs at the end of the horizon. The heir’s marginal income tax rate on inherited tax-deferred balances can be specified by the user.

Solver options for spending and bequest

Two options modify how the primary objectives are formulated:

- **timePreference** (%/year, default 0) — a subjective time-discount rate ρ that front-loads spending. The objective weight for year n is multiplied by $(1+\rho)^{-n}$, so near-term spending is valued more than equivalent spending in the distant future. At $\rho = 0$ (the default) all years carry equal real weight.
- **fixedSpending** (monetary value) — pins first-year net spending to a user-specified level when using **maxSpending**. Instead of freely maximizing the basis, the optimizer holds year-0 spending fixed and uses a slack variable to allow the spending profile to flex dynamically around that anchor in subsequent years. Useful for testing a specific withdrawal rate or matching a concrete budget target.

Lexicographic weight

A small secondary weight can be added to the objective to break ties among solutions with the same primary objective value. By default, this weight mildly discourages unnecessary Roth conversions (preferring fewer conversions when spending is otherwise unaffected) and mildly prefers drawing withdrawals from the first individual before the second when both choices are equivalent. The weight is small enough that it never overrides the primary objective.

3.6 The Self-Consistent Iteration Loop

Several quantities that enter the LP constraints are nonlinear functions of the income variables being optimized. The most important are:

- the *Social Security taxability fraction*—the share of SS benefits subject to ordinary income tax—which depends on provisional income, which in turn includes SS benefits themselves and all other income sources;

- *Medicare IRMAA costs*, which depend on MAGI from two years prior and interact with the current year's spending and tax calculations;
- the *Net Investment Income Tax (NIIT)*, which depends on MAGI and on investment income;
- the *LTCG bracket upper bounds*, which depend on how much of each threshold is already occupied by ordinary income;
- the *OBBA 65+ senior bonus deduction phaseout*, which is a nonlinear function of MAGI.

Owl handles these nonlinearities through the SC loop. At each iteration, all nonlinear quantities are held fixed at their values from the previous iteration, making the problem an LP (or MIP). The solver produces a new income solution; the nonlinear quantities are recomputed from that solution and compared to their previous values. If the maximum change across all years falls below a convergence tolerance, the loop terminates; otherwise the updated values are substituted and the process repeats.

To prevent oscillation near the SS taxability thresholds—where small changes in income can flip the taxability fraction back and forth—the updated SS taxability estimate is blended with the previous estimate using a damped update. The loop typically converges in fewer than fifteen iterations for well-posed plans.

Five optional MIP formulations replace portions of the SC loop with exact binary constraints, eliminating the corresponding nonlinearity entirely:

- `withSSTaxability="optimize"` — models SS taxability as an exact piecewise-linear MIP using binary variables z^s to select the correct taxability threshold (0%, 50%, or 85%) each year.
- `withLTCG="optimize"` — uses binary variables z^l to select the correct 0%, 15%, or 20% LTCG bracket each year, rather than approximating bracket thresholds from the previous SC iteration.
- `withNIIT="optimize"` — uses a binary variable z^j to determine whether MAGI exceeds the NIIT threshold each year, and a continuous surplus variable δ^j to enforce the IRS NII cap ($\min(\mathbb{G} - \mathbb{G}^{\text{NIIT}}, \mathbb{I} + Q)$) without a second binary.
- `withACA="optimize"` — folds ACA premium bracket selection and MAGI into the MIP, replacing the per-iteration SC approximation.
- `withMedicare="optimize"` — models IRMAA bracket selection using binary variables z^m , guaranteeing the globally optimal bracket assignment each year.

Each exact formulation eliminates the corresponding SC-loop approximation at the cost of additional binary variables. When multiple flags are active simultaneously, the resulting MILP can be large; the `withDecomposition` option (Section 3.9) provides sequential (relax-and-fix) and Benders strategies to reduce solve time.

3.7 Filing Status and the Death of a Spouse

For married couples, the year of the first spouse’s death is a known input to the plan. Beginning in that year, Owl automatically transitions all relevant quantities from married-filing-jointly (MFJ) status to single status:

- the standard deduction drops from the MFJ amount to the single-filer amount;
- tax bracket widths narrow to the single-filer schedule;
- Social Security taxability thresholds shift to the single-filer values;
- Medicare premiums reflect only the surviving spouse;
- the survivor begins receiving the survivor benefit computed at the year of the spouse’s death;
- the spending profile is multiplied by the user-specified survivorship factor.

Because the death year is fixed at plan-construction time, this transition requires no binary variables. Separate parameter vectors are constructed for the MFJ period and the single-survivor period; the LP selects the correct values automatically for each year.

3.8 Solvers

Owl’s optimization model is assembled in a solver-agnostic matrix form: inequality constraints, equality constraints, variable bounds, and an objective vector. Any LP or MIP solver that accepts this standard input format can be used.

The default solver is **HiGHS**, a high-performance open-source LP and MIP solver accessed directly via the `highspy` Python package. An optional interface to **MOSEK**, a commercial solver requiring a separate license, is also supported and generally provides faster solution times for large MIP problems.

3.9 MIP Decomposition

When multiple "optimize" flags are active simultaneously—for example, `withLTCG`, `withMedicare`, `withACA`, `withSSTaxability`, and `withNIIT` all set to "optimize"—the monolithic MILP can accumulate on the order of 300 binary variables from the five bracket-selector families ($z^m, z^a, z^\sigma, z^l, z^j$). These are the only binaries a plan carries; with every tax mode left in loop mode the problem is a pure linear program. Solve times grow substantially with this binary count. The `withDecomposition` option partitions the MILP into two interacting subproblems that are solved sequentially, separating bracket selection from continuous planning:

- **Master problem** — selects the bracket binary families $z^m, z^a, z^\sigma, z^l, z^j$.
- **Subproblem** — solves the continuous planning problem (all LP variables), with bracket assignments fixed by the master.

Two decomposition strategies are available.

Sequential (relax-and-fix). The LP relaxation (all binaries relaxed to $[0, 1]$) is solved first. The bracket binary families are then rounded to 0 or 1 in a fixed order ($z^l \rightarrow z^\sigma \rightarrow z^j \rightarrow z^m \rightarrow z^a$). Finally, the reduced problem is solved with all bracket binaries fixed. This is a fast heuristic and is not guaranteed to be globally optimal, but in practice it produces near-optimal or optimal solutions.

Benders decomposition. Classical Benders decomposition [38] provides a certified global optimum. Starting from an initial bracket assignment, the subproblem LP is solved and its dual variables π are used to generate a Benders optimality cut

$$\eta \geq \alpha + \beta^\top z, \quad \alpha = \theta_{\text{LP}} - \beta^\top z^*, \quad \beta_j = - \sum_i \pi_i A_{ij}, \quad (3.1)$$

where θ_{LP} is the subproblem LP objective and A_{ij} are the constraint coefficients of master column j . The cut is added to the master; the master MIP is re-solved for a new bracket assignment and a lower bound on η . An upper bound is obtained by solving the subproblem MIP. The loop terminates when the gap between the upper and lower bounds falls within the solver's MIP gap tolerance. In practice the algorithm converges in one to three iterations. If the LP relaxation gap cannot be closed, Benders falls back to the sequential strategy.

Both strategies support HiGHS and MOSEK. The option is `withDecomposition="none"|"sequential"|"benders"`; the default is `"none"` (monolithic MILP).

4. Indices, variables, and parameters

In the next sections, the indices, variables, and parameters are described in detail. Then the model constraints are introduced. For implementation in a linear programming solver, index mapping functions are introduced to map all variables into a single one-dimensional array that is optimized subject to inequality and equality constraints expressed in matrix form. Finally, the constraint matrices are built and so are some useful objective functions.

4.1 Indices

For all indices, we will follow the C array style (starting at 0), rather than the traditional mathematical standard starting at 1. This will facilitate the final sequential mapping of all the variables into a single one-dimensional array, and serve as a direct reference for better understanding the code implementation.

The indices used and their range are defined here, while we also introduce the characteristics and dimensions of the problem. Upper bounds on indices are indicated by the letter N , with the index name as a subscript, e.g., N_i for index i . In other instances, the subscript will indicate that the array depends on subscripts, for example, b_{ikn} is a multidimensional array depending on subscripts i , k , and n .

- i Individual. i runs from 0 to $N_i - 1$ where $N_i = 2$ for couples, or $N_i = 1$ for single individuals. The first individual to pass is denoted by i_d while the survivor is i_s .
- j Type of savings account. j goes from 0 to $N_j - 1$, for taxable ($j = 0$), tax-deferred ($j = 1$), tax-free Roth ($j = 2$), and Health Savings Account ($j = 3$) respectively. Therefore $N_j = 4$.

- k Type of asset class. k goes from 0 to $N_k - 1$, for S&P 500, Baa corporate bonds, Treasury notes, and cash, respectively, and therefore $N_k = 4$. More asset classes could be considered at the cost of increasing the complexity of the problem while not generating many more insights.
- n Index of the year being modeled. The period being modeled runs from the beginning of year 0 to the first day of the year following year $N_n - 1$, and therefore $N_n + 1$ years are actually considered. Year N_n is the first year following the passing of all individuals in the plan. The time period for all decision variables is annual. For spouses, the end of year $n_d - 1$ is when the first individual is assumed to pass while the survivor is assumed to die at the end of year $N_n - 1$ of the plan. $n_{m,i}$ denotes the first year individual i is on Medicare (turns 65), so that $n_m = \min_i n_{m,i}$ is the first year any individual is on Medicare and no Medicare cost is incurred for $n < n_m$. $n_{aca} = \max_i \min(n_{m,i}, N_n^{(i)})$, where $N_n^{(i)}$ is the planning horizon of individual i , is the first year no individual is ACA-eligible; ACA applies for $n < n_{aca}$. For a single individual $n_m = n_{aca}$; for a couple $n_m \leq n_{aca}$, with both Medicare and ACA costs potentially applying in the transition years $n_m \leq n < n_{aca}$.
- p Index for long-term capital gains (LTCG) tax rate brackets. $p \in \{0, 1, 2\}$ corresponds to the 0%, 15%, and 20% LTCG rates, respectively, giving $N_p = 3$ brackets.
- q Index for income brackets related to Medicare premium adjustments. q goes from 0 to $N_q - 1$, from low to high. At the time of writing, there are 5 Medicare premium step adjustments in the federal tax code separating $N_q = 6$ income brackets.
- r ACA MAGI bracket index. r goes from 0 to $N_r - 1$ with $N_r = 7$ brackets (6 FPL-based intervals up to 400% FPL, plus one above).
- t Federal income tax bracket. t goes from 0 to $N_t - 1$, from low to high. At the time of writing, there are $N_t = 7$ federal income tax brackets in the current federal tax code.

4.2 Variables

We will use lowercase roman letters to represent variables. This is done to separate variables from parameters, which will be represented using Greek letters or

caligraphic fonts. All variables are assumed to take only non-negative values (≥ 0 inequality). Variables are resolved by the optimizer while parameters are prescribed by the user, historical data, or tax laws.

- b_{ijn} Balance for individual i in savings account j at the beginning of year n . When we consider each asset class k , the variable b_{ijkn} is used instead.
- d_{in} Deposit of year- n net spending surplus in taxable account of individual i . These deposits are coming from the surplus s_n , distributed to spousal taxable accounts depending on parameter η .
- e_n Adjusted standard exemption for year n . This is a variable as the taxable income can sometimes be less than the standard exemption $\bar{\sigma}_n$, leading to a negative taxable income if the inflation-adjusted standard exemption is simply subtracted from the gross taxable income G_n in years of low income.
- f_{tn} Amount of taxable ordinary income falling into tax bracket t . The ordinary taxable income is

$$G_n = \sum_{t=0}^{N_t-1} f_{tn}, \quad (4.1)$$

with f_{tn} bound between 0 and the bracket width $\bar{\Delta}_{tn} = \bar{\Gamma}_{tn} - \bar{\Gamma}_{(t-1)n}$. For interpretation, one can introduce the fractional fill f'_{tn} with $f_{tn} := f'_{tn} \bar{\Delta}_{tn}$ and

$$G_n = \sum_{t=0}^{N_t-1} f'_{tn} \bar{\Delta}_{tn}, \quad (4.2)$$

$$0 \leq f'_{tn} \leq 1. \quad (4.3)$$

When considering taxes, the product $\bar{\Delta}_{tn} \theta_{tn}^x$ need not be ordered monotonically, so we cannot rely on the cost structure to fill lower brackets first when minimizing. It is therefore more appropriate to optimize directly in f_{tn} , as the tax rates are progressive. Given that the rates θ_{tn}^x on tax brackets t are increasing monotonically with income, the lower brackets will be filled in first when optimizing. Definitions of θ_{tn}^x , Γ_{tn} and Δ_{tn} are in the section describing the parameters below.

- g_n Net spending in year n .

h_{qn} Segment of Medicare bracket q in which the modified adjusted gross income (MAGI) in year $n - 2$ is contained. This variable is used to model the income-related monthly adjustment amounts (IRMAA) brackets using special ordered sets of type 1 (SOS1) for the selection on the bracket indicators (as opposed to using a big- $\mathcal{M}$ method). One of the brackets, say q' , will contain the MAGI from two years before $\mathbb{G}_{(n-2)}$, and therefore

$$h_{qn} = \delta_{q,q'} \mathbb{G}_{(n-2)}, \quad (4.4)$$

where $\delta_{q,q'}$ is the Kronecker delta (1 if $q = q'$, 0 otherwise), such that

$$\mathbb{G}_{(n-2)} = \sum_q h_{qn}. \quad (4.5)$$

The role of h will become clearer as the formulation of the MAGI calculation is exposed below.

m_n Medicare costs in year n , including Part B and Part D premiums and IRMAA income-related adjustments for both parts. Part D can be disabled via the `includeMedicarePartD=false` option. In `withMedicare="optimize"` mode, m_n is an LP variable determined within the MIP. In `withMedicare="loop"` mode (the default), m_n is computed via the SC loop and treated as a parameter.

h_{rn}^{aca} Segment of ACA bracket r in which the current-year modified adjusted gross income (MAGI) $\mathbb{G}_n^{\text{aca}}$ is contained for ACA-eligible year n . Used only when `withACA="optimize"`; index n runs over the first n_{aca} plan years when at least one individual is under age 65 and within their horizon. One bracket r' contains MAGI, so $\sum_r h_{rn}^{\text{aca}} = \mathbb{G}_n^{\text{aca}}$. ACA uses *current-year* MAGI (no two-year lag like Medicare IRMAA).

m_n^{aca} ACA net premium cost in year n (after Premium Tax Credit). Used only when `withACA="optimize"`; zero for $n \geq n_{\text{aca}}$ (Medicare-eligible years). In `withACA="loop"` mode, ACA cost is computed via the SC loop and treated as a parameter.

s_n Surplus of funds during year n , most likely caused by required minimum distributions (RMDs) or influx of money from big-ticket items (inheritance, gifts, etc.), or from the disposition of fixed assets (sale of residence, restricted stocks, etc.).

w_{ijn} Withdrawal from account j belonging to individual i at the beginning of year n . For the ($j = 1$) tax-deferred savings account, w_{i1n} is referred to as a distribution for tax purposes as it is a taxable withdrawal, and will always satisfy required minimum distributions. For the ($j = 3$) HSA, w_{i3n} represents qualified medical withdrawals, which are entirely tax-free and excluded from taxable income (simplified model; see Section 2.8).

x_{in} Roth conversion performed by individual i during year n . These events are taxable as ordinary income. Roth conversions are forced to zero in the last two years of each individual's horizon (see Limitations in Section 2.7).

$p_n^{\sigma, \text{lo}}$,
 $p_n^{\sigma, \text{hi}}$ Excess provisional income above the lower and upper Social Security taxability thresholds, respectively:

$$p_n^{\sigma, \text{lo}} = \max(0, \Pi_n - \mathcal{P}_n^{\text{lo}}), \quad (4.6)$$

$$p_n^{\sigma, \text{hi}} = \max(0, \Pi_n - \mathcal{P}_n^{\text{hi}}). \quad (4.7)$$

These continuous non-negative variables are present only when the associated solver option `withSSTaxability="optimize"`. See Eq. (5.48) for the formal constraint formulation.

$p_n^{\sigma, \text{min}}$ 50%-zone Social Security taxability allocation, bounded in $[0, \Delta\mathcal{P}]$, where $\Delta\mathcal{P} := \mathcal{P}_n^{\text{hi}} - \mathcal{P}_n^{\text{lo}}$ is the 50%-zone width (see Parameters):

$$p_n^{\sigma, \text{min}} = \min(\Delta\mathcal{P}, p_n^{\sigma, \text{lo}}). \quad (4.8)$$

Only present when `withSSTaxability="optimize"` is selected. Together with $p_n^{\sigma, \text{lo}}$ and $p_n^{\sigma, \text{hi}}$, all SS taxability LP variables use the letter p for consistency.

q_{pn} Portion of LTCG and qualified dividends allocated to bracket p in year n , where $p = 0, 1, 2$ correspond to the 0%, 15%, and 20% LTCG rate brackets, respectively (the leading letter q here is *not* the Medicare IRMAA bracket index q from the Indices list; the LTCG bracket index is p). These $N_p = 3$ continuous non-negative LP variables are always present and together partition the total LTCG:

$$q_{0n} + q_{1n} + q_{2n} \geq Q_n, \quad (4.9)$$

$$q_{0n} \leq \max(0, \bar{\mathcal{B}}_n^{15} - G_n), \quad (4.10)$$

$$q_{0n} + q_{1n} \leq \max(0, \bar{\mathcal{B}}_n^{20} - G_n), \quad (4.11)$$

where $\bar{\mathcal{B}}_n^{15}$ and $\bar{\mathcal{B}}_n^{20}$ are the inflation-adjusted total taxable income thresholds above which the 15% and 20% rates apply (see Parameters), and $G_n = \sum_{t=0}^{N_t-1} f_{tn}$ is the ordinary taxable income (formally defined in the Intermediate Variables section). Since the cost function is strictly convex ($0 < 0.15 < 0.20$), no binary variables are required: the LP naturally fills the lowest-cost bracket first. See Eqs. (5.29)–(5.30) for the SC-loop formulation. When `withLTCG="optimize"` is selected, G_n becomes a continuous LP variable and binary variables z_{0n}^l, z_{1n}^l replace the SC-loop approximation of bracket thresholds with exact big- $\mathcal{M}$ constraints (see Eqs. (5.31)–(5.35)).

t_n^σ Taxable Social Security amount in year n , bounded in $[0, 0.85 \sum_i \bar{\zeta}_{in}]$. In the default SC-loop mode this quantity equals $R_n \sum_i \bar{\zeta}_{in}$ and is treated as a fixed parameter; with the solver option `withSSTaxability="optimize"` it becomes an LP decision variable computed exactly from the IRS formula without iteration.

z_*^* Binary variables are all designated by the letter z ; a superscript distinguishes the five families:

- z_{qn}^m — Medicare IRMAA bracket selectors; present when `withMedicare="optimize"`.
- z_{rn}^{aca} — ACA premium bracket selectors; present when `withACA="optimize"`.
- $z_{0n}^\sigma, z_{1n}^\sigma$ — encode the two $\min(\cdot, \cdot)$ operations in the SS taxability MIP; present when `withSSTaxability="optimize"`.
- z_{0n}^l, z_{1n}^l — LTCG bracket-room indicators ($z^l = 1$ iff ordinary income G_n is below the 15% or 20% threshold); present when `withLTCG="optimize"`.
- z_n^j — NIIT threshold indicator ($z^j = 1$ iff MAGI exceeds $\mathbb{G}^{\text{NIIT}}$); present when `withNIIT="optimize"`.
- δ_n^j — continuous NII surplus, $\delta_n^j = \max(0, \mathbb{G}_n - \mathbb{G}^{\text{NIIT}} - \mathbb{I}_n - Q_n)$; present when `withNIIT="optimize"`. Driven to its upper bound by the minimizing optimizer to enforce the IRS NII cap without a second binary.

See the corresponding constraint subsections for the big- $\mathcal{M}$ formulations.

4.3 Parameters

For more easily distinguishing parameters from variables, all parameters are expressed either in Greek letters or uppercase roman letters using caligraphic fonts.

Parameter values are either set by the user, historical data, or by the tax code.

β_{ij} Initial balances in savings accounts. These amounts are used to initialize b_{ij0} .

τ_{kn} Annual rate of return for asset class k in year n . A time series of annual return rates for each asset class. Here, inflation and the rate of return of cash ($k = 3$) are assumed to be the same. In other words, investing in cash yields constant dollars as the return perfectly matches inflation, equivalent to a Treasury inflation-protected security (TIPS) having a real yield of 0%.

γ_n Cumulative inflation at the beginning of year n computed as the product

$$\gamma_n = \prod_{n'=0}^{n-1} (1 + \tau_{3n'}), \quad (4.12)$$

with $\gamma_0 := 1$, and where n' is a dummy index. As the time span of interest goes from the beginning of the first year to the beginning of the year following the last year, variable γ_n will have $N_n + 1$ elements. Parameters indexed for inflation will be indicated by a bar on top as in $\bar{\sigma}_n$. See the next entry for a specific example.

ψ_{in} Effective capital-gain coefficient for withdrawals from person i 's taxable account in year n . It is computed by the self-consistent loop and enters the LP as a fixed parameter each iteration. When a cost basis is provided via `setCostBasis()`, $\psi_{in} = 1 - K_{in}/b_{i0n}$ (see Eq. (4.27)). Otherwise, $\psi_{in} = \max(0, \tau_{0(n-1)} - \mu)$, i.e., only the most recent year's price appreciation is treated as a gain (Eq. (4.25)).

σ_n Standard deduction. It can be adjusted for inflation as follows

$$\bar{\sigma}_n = \sigma_n \gamma_n, \quad (4.13)$$

and can be modified for additional exemptions after age 65, for example. It is a simple time series which can include any foreseeable changes in the tax code, or change in filing status due to the passing of one spouse for $n \geq n_d$. The value of $\bar{\sigma}_n$ is an upper bound for variable e_n .

ξ_n Spending profile. This is a time series that multiplies a basis for the desired net spending amount. It is $\xi_n = 1, \forall n$ for a flat profile, or can be a *smile* profile allowing for more money at the start of retirement and modulating

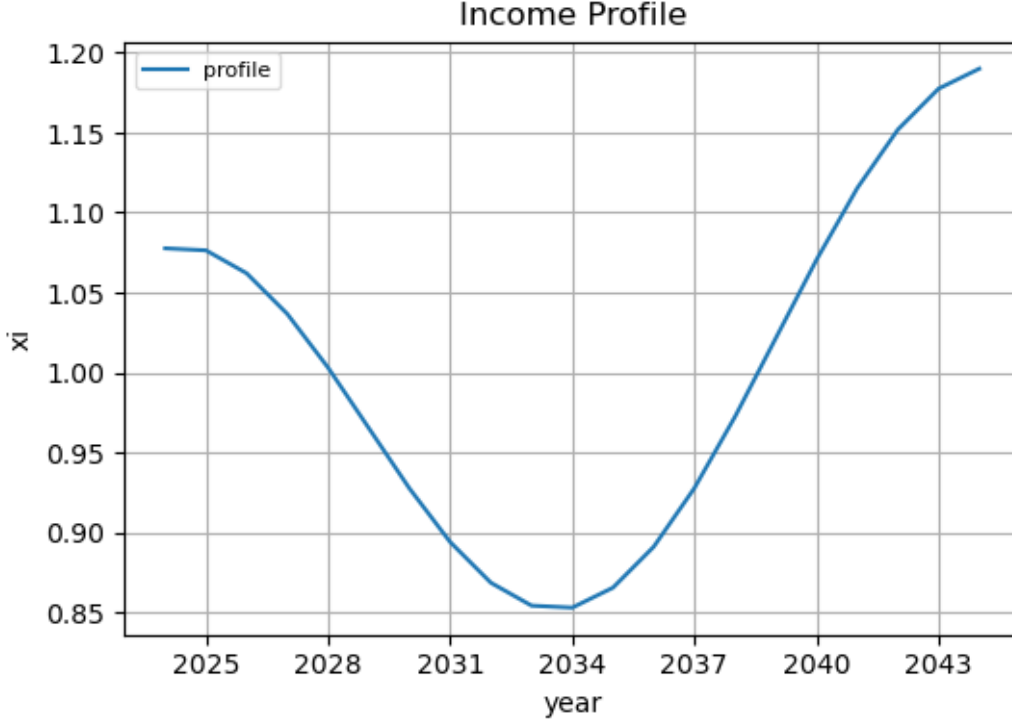


Figure 4.1: Example of a spending profile with a 15% cosine factor and a 12% linear component.

it over retirement. Parameter ξ_n can also contain spending adjustments typically made at the passing of one spouse. The *smile* can be implemented using a cosine superimposed over a gentle linear increase such as in

$$\xi_n = 1 + a_1 * \cos(2n\pi/(N_n - 1)) + a_2 n/(N_n - 1), \quad (4.14)$$

and then normalized by factor $N_n/(\sum_n \xi_n)$ to be sum-neutral with respect to a flat profile. Values of $a_1 = 15\%$ and $a_2 = 12\%$ provide curves that are similar to realistic spending profiles reported in the literature. See Fig. 4.1 for an example. At the passing of one spouse, both profiles are reduced by a factor χ for $n \geq n_d$, and the normalizing factor is adjusted accordingly. The inflation-adjusted profile is $\bar{\xi}_n = \gamma_n \xi_n$.

χ Factor to reduce spending profile after the passing of one spouse. It is typically assumed to be 0.6. That is, we are assuming that the surviving

spouse can live with 60% of the net spending amount that was available to the couple.

- n_d Year index at which the first spouse (i_d) is assumed to pass. For single individuals, $n_d = N_n$. For couples, n_d is derived from the life-expectancy inputs so that the end of year $n_d - 1$ coincides with the assumed date of death of i_d . Filing status, tax brackets, Social Security thresholds, and LTCG thresholds all change at $n = n_d$ (from married filing jointly to single).
- $n_{m,i}$, n_m $n_{m,i}$ is the year index at which individual i turns 65 and joins Medicare. $n_m = \min_i n_{m,i}$ is the first year any individual is on Medicare; no Medicare cost m_n is incurred for $n < n_m$. Used in the Medicare constraints; see the Medicare paragraph in the Constraints section.
- $\mathcal{N}_i^{\text{hsa}}$ Year index at which individual i enrolls in Medicare and HSA contributions must cease. By default $\mathcal{N}_i^{\text{hsa}} = n_{m,i}$ (age 65), but `setHSA()` accepts a custom enrollment age so the two can differ. For $n \geq \mathcal{N}_i^{\text{hsa}}$, the contribution κ_{i3n} is zero; values in the *HSA ctrb* column for those years are ignored.
- ρ_{in} Required minimum distribution for individual i in year n . Expressed in fractions determined from IRS tables. The IRS Uniform Lifetime Table III (Pub. 590-B) is used in standard cases. When the spouse is the sole designated beneficiary and more than ten years younger than the account owner, the Joint and Last Survivor Table II is used instead.
- Γ_{tn} Bounds for federal income tax brackets. We define $\Gamma_{(-1)n} := 0$, so that Γ_{0n} is the upper bound for the 10% tax bracket in year n . As the filing status can change for couples, and so can the tax code, Γ_{tn} will be changing over n . Inflation-adjusted brackets are $\bar{\Gamma}_{tn} = \gamma_n \Gamma_{tn}$.
- Δ_{tn} Difference between upper bound Γ_t and lower bound Γ_{t-1} of a federal income tax bracket,

$$\Delta_{tn} = \Gamma_{tn} - \Gamma_{(t-1)n}. \quad (4.15)$$

Inflation-adjusted bracket widths are $\bar{\Delta}_{tn} = \gamma_n \Delta_{tn}$. Once adjusted for inflation, the taxable income can be expressed as in Eq. (4.1). These data are 7 time series. The filing status changes after the passing of one spouse ($n \geq n_d$) and income tax brackets and differences are adjusted accordingly.

- θ_{tn}^x Tax rate for ordinary income tax bracket t in year n . Using N_t time series allows one to adjust income tax rates in the foreseeable future. For example, as of 2026 the rates (in decimal) are .10, .12, .22, .24, .32, .35, and .37. While these rates were extended indefinitely by Congress in 2025, they could still revert in the future to 2017 or similar higher rates (.10, .15, .25, .28, .33, .35, and .396). See Eq. (4.35) for its use.
- $\mathcal{B}^{15}, \mathcal{B}^{20}$ Base (2026 nominal) total taxable income thresholds above which the 15% and 20% long-term capital gains (LTCG) tax rates apply, respectively. LTCG is taxed at 0% when total taxable income (ordinary + LTCG) is below $\bar{\mathcal{B}}_n^{15}$, at 15% between $\bar{\mathcal{B}}_n^{15}$ and $\bar{\mathcal{B}}_n^{20}$, and at 20% above $\bar{\mathcal{B}}_n^{20}$, where $\bar{\mathcal{B}}_n^{15} = \gamma_n \mathcal{B}^{15}$ and $\bar{\mathcal{B}}_n^{20} = \gamma_n \mathcal{B}^{20}$ are the inflation-adjusted thresholds. In 2026, $(\mathcal{B}^{15}, \mathcal{B}^{20}) = (\$49,450, \$545,500)$ for single filers and $(\$98,900, \$613,700)$ for married filing jointly. Filing status changes at $n = n_d$ for couples. Used in the LTCG bracket constraints; see Eqs. (5.29)–(5.30).
- α_{ijkn} Desired asset allocation for savings account j of individual i in asset class k during year n . Allocation ratios come in many flavors as they could be specified globally between individuals and accounts as α_{kn} , for example. When specified by the user, allocation ratios are given two values, one at the beginning of the plan α_{ijk0} and the other at the end $\alpha_{ijkN_{n-1}}$, or α_{ijkn_d} for a spouse passing before the other. Then, intermediate values are interpolated either using a linear relation,

$$\alpha_{ijkn} = a + \frac{n}{N-1}(b-a), \quad (4.16)$$

where N is either n_d or N_n , or using an s-curve as in

$$\alpha_{ijkn} = a + \frac{(b-a)}{2}(\tanh((n-n_1)/n_2) + 1), \quad (4.17)$$

where n_1 is the number of years ahead when the inflection point will occur, and n_2 is the width (in years) of the transition. Constants n_1 and n_2 can be adjusted by the user. Default values are $n_1 = 15$, and $n_2 = 5$, meaning that the transition center will occur in 15 years, taking place from 15 – 5 years to 15 + 5 years from now. Using $a = \alpha_{ijk0}$ and $b = \alpha_{ijk(N-1)}$ is an approximation as values of ± 1 are only reached at $\pm \infty$ for a hyperbolic tangent. More precise bounds a' and b' for matching the desired start and end values can be determined by solving a 2×2 system of equations leading

to

$$\begin{aligned} a' &= (a - k_{12}b')/k_{11} \\ b' &= (b - (k_{21}/k_{11})a)/(k_{22} - (k_{21}/k_{11})k_{12}), \end{aligned} \quad (4.18)$$

where

$$\begin{aligned} k_{11} &= \frac{1}{2}(1 + \tanh(n_1/n_2)) \\ k_{12} &= \frac{1}{2}(1 - \tanh(n_1/n_2)) \\ k_{21} &= \frac{1}{2}(1 - \tanh((N - 1 - n_1)/n_2)) \\ k_{22} &= \frac{1}{2}(1 + \tanh((N - 1 - n_1)/n_2)). \end{aligned} \quad (4.19)$$

These interpolation functions allow the allocation ratios to gradually change or *glide* during retirement. Fig. 4.2 provides an example of an *s-curve* gliding allocation ratios.

It is also possible to have a coarser granularity on the portfolio by having an asset allocation scheme defined on a sum of accounts. For example, allocation can be coordinated between accounts leading to α_{ikn} , or even between spouses as α_{kn} . For any of these cases, it is assumed that weights are always properly scaled so that

$$\begin{aligned} \sum_k \alpha_{ijkn} &= 1, \\ \text{or} \quad \sum_k \alpha_{ikn} &= 1, \\ \text{or} \quad \sum_k \alpha_{kn} &= 1, \end{aligned} \quad (4.20)$$

depending on the scheme selected.

$\mathcal{T}_{ijn}$

When the allocation ratios α_{ijkn} are prescribed, it is sometimes more convenient to express the return rates as

$$\mathcal{T}_{ijn} = \sum_k \alpha_{ijkn} \tau_{kn}. \quad (4.21)$$

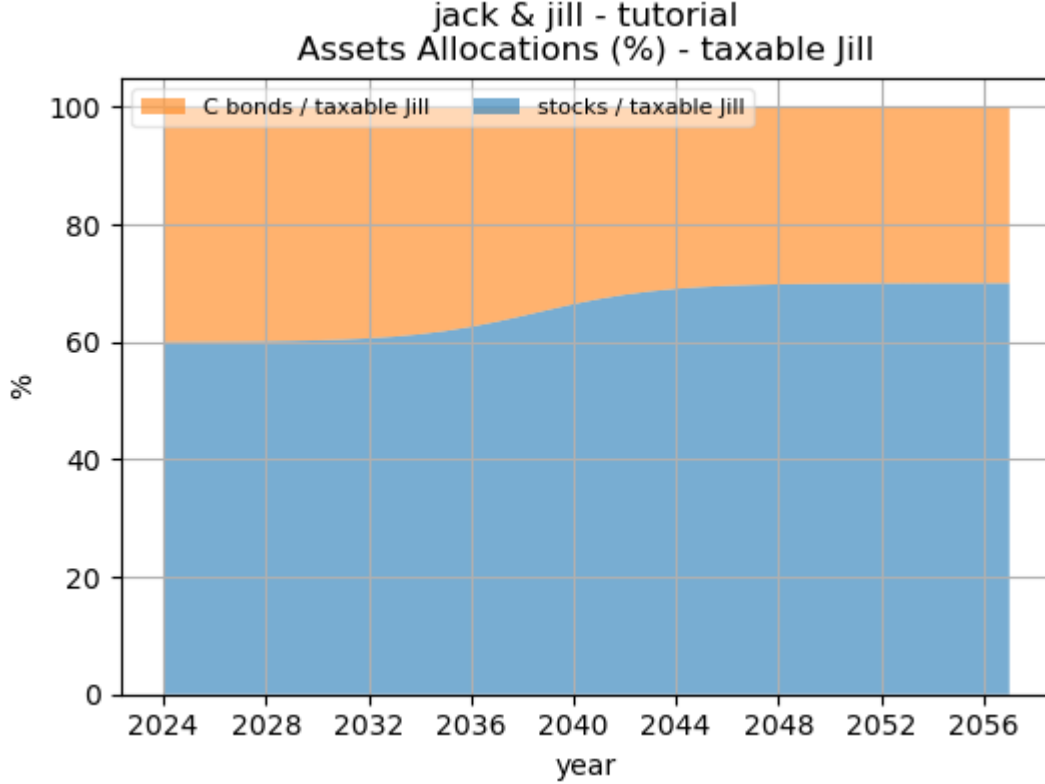


Figure 4.2: Example of an allocation portfolio with 60/40% stocks/bonds transitioning to 70/30% using an s-curve.

- $\Lambda_{in}^{\pm}$ Big-ticket item requested by individual i in year n . These are large expenses or influx of money that can be planned. Therefore, $\Lambda^{\pm}$ can be positive (e.g., gift received, inheritance) or negative (e.g., buy a house, large gifts).
- λ Allowed deviation from the desired net spending profile during one year. Parameter λ can be better understood as a percentage. If $\lambda = 0.10$, then the net spending amount is allowed to vary by up to 10% from the prescribed profile. This parameter is mainly provided for educational purposes.
- π_{in} Sum of pension benefits for individual i in year n . These amounts are typically specified along with the ages at which these benefits begin. Pensions can optionally be indexed for inflation and then represented as $\bar{\pi}_{in}$.
- ζ_{in} Social Security benefits for individual i in year n . Starting age and the

passing of one individual for spouses will determine the time series. $\bar{\zeta}_{in}$ is the same series adjusted for inflation.

$\mathcal{P}_n^{\text{lo}}, \mathcal{P}_n^{\text{hi}}, \Delta\mathcal{P}$ Lower and upper provisional income thresholds for Social Security taxability, following IRS Publication 915. Below $\mathcal{P}_n^{\text{lo}}$ no Social Security is taxable; between $\mathcal{P}_n^{\text{lo}}$ and $\mathcal{P}_n^{\text{hi}}$ up to 50% of benefits enter taxable income; above $\mathcal{P}_n^{\text{hi}}$ up to 85% of benefits are taxable. We define $\Delta\mathcal{P} := \mathcal{P}_n^{\text{hi}} - \mathcal{P}_n^{\text{lo}}$ (the 50%-zone width; \$9,000 for single, \$12,000 for MFJ). For single filers $(\mathcal{P}^{\text{lo}}, \mathcal{P}^{\text{hi}}) = (\$25,000, \$34,000)$; for married filing jointly $(\mathcal{P}^{\text{lo}}, \mathcal{P}^{\text{hi}}) = (\$32,000, \$44,000)$. Unlike income-tax brackets, these thresholds are *not* indexed for inflation; the time subscript n reflects only the change in filing status at $n = n_d$.

$\mathbb{G}^{\text{NIIT}}$ Statutory modified adjusted gross income (MAGI) threshold above which the net investment income tax (NIIT) of 3.8% applies. $\mathbb{G}^{\text{NIIT}}$ is not indexed for inflation. In 2026, $\mathbb{G}^{\text{NIIT}} = \$200,000$ for single filers and \$250,000 for married filing jointly, with filing status changing at $n = n_d$. See the definition of J_n in the Intermediate Variables section.

ϵ_{N_n} Desired amount to leave as a bequest at the end of the final year of the plan, $N_n - 1$, which is the beginning of year N_n . This amount is the after-tax value of the estate for the heirs expressed in today's dollars. The inflation-adjusted target is $\bar{\epsilon}_{N_n} = \gamma_{N_n} \epsilon_{N_n}$. See parameter ν for the heirs tax rate.

κ_{ijn} Sum of contributions to savings account j made by individual i during year n . We assume that contributions are made at half-year to better represent periodic contributions made throughout the year. In practice, a contribution amount κ_{ijn} is specified, in which case the contribution to each asset class is

$$\kappa_{ijkn} = \alpha_{ijkn} \kappa_{ijn}. \quad (4.22)$$

as savings account balances are assumed to be rebalanced periodically. For the $(j = 3)$ HSA, contributions κ_{i3n} are pre-tax: as with all contributions, wages ω_{in} are entered net of these amounts, which excludes them from taxable income, provisional income for Social Security taxability, and MAGI for Medicare IRMAA purposes in year n . HSA contributions are zeroed for $n \geq \mathcal{N}_i^{\text{hsa}}$, when individual i enrolls in Medicare (typically at age 65).

ω_{in} Sum of wages earned by individual i during year n , net of all account contributions (see Section 2.3). Do not confuse wages ω with withdrawals w .

v_{in}	Other ordinary income for individual i in year n , beyond wages, pension, and Social Security. This is a user-supplied time series (e.g., rental income, alimony) loaded from the household financial plan spreadsheet.
ν_{in}	Net investment income from rent or trust distributions for individual i in year n . User-supplied time series loaded from the <code>netinv</code> column of the household financial plan spreadsheet. It enters cash-flow and taxable-income constraints as ordinary income and is also included in the net investment income total for NIIT computation. Defaults to zero when the column is absent.
$\mathcal{A}_n^*$	Amounts resulting from the proceeds of liquidation of fixed assets. The liquidation of these fixed assets can generate a taxable income, marked with a superscript x as $\mathcal{A}_n^x$, long-term capital gains, marked with superscript c as $\mathcal{A}_n^c$, or tax-free proceeds marked with a superscript f as $\mathcal{A}_n^f$. These are used to capture the sale of a residence or real estate, cashing restricted stocks, or receiving a lump-sum annuity.
$\mathcal{C}_q$	Combined Medicare Part B and Part D cost for IRMAA bracket q of modified adjusted gross income (MAGI) $\mathbb{G}$, including base premiums and income-related monthly adjusted amounts (IRMAA) for both parts. When adjusted for inflation, this becomes $\bar{\mathcal{C}}_{qn} = \gamma_n \mathcal{C}_q$. There are $N_q = 6$ brackets for IRMAA and therefore 5 step adjustments forming a piecewise constant function.
$\mathcal{D}_n$	Debt payment in year n to be considered in the cash flow. This allows the cash flow to account for a mortgage or a car loan, for example.
$\mathcal{L}_q$	Upper bounds for brackets used to determine Medicare adjustments based on the modified adjusted gross income (MAGI) $\mathbb{G}$. When adjusted for inflation, this becomes $\bar{\mathcal{L}}_{qn} = \gamma_n \mathcal{L}_q$. While there are 6 brackets for IRMAA adjustments forming a piecewise constant function, $\mathcal{L}_q$ has $N_q - 1$ distinct thresholds as the last element is an arbitrarily large number bounding the last bracket. See Fig. 5.1 for a visual representation. Note that $\bar{\mathcal{L}}_{qn}$ needs to be adjusted for inflation and marital status, including adjustments due to the passing of one spouse.
$\bar{\mathcal{L}}_{rn}^{\text{aca}},$ $\bar{\mathcal{C}}_r^{\text{aca}},$ $\bar{\mathcal{S}}_n^{\text{aca}}$	ACA parameters (only when <code>withACA="optimize"</code> and $n < n_{\text{aca}}$). $\bar{\mathcal{L}}_{rn}^{\text{aca}}$: inflation-adjusted FPL bracket thresholds for year n ; $N_r = 7$ brackets (6 FPL-based intervals up to 400% FPL, plus one above). $\bar{\mathcal{C}}_r^{\text{aca}}$: applicable

contribution percentage for bracket r ($r = 0, \dots, 5$); contribution is proportional to MAGI within each bracket. $\bar{S}_n^{\text{aca}}$: inflation-adjusted annual benchmark Silver plan (SLCSP) premium; for bracket 6 (above 400% FPL), net cost equals $\bar{S}_n^{\text{aca}}$ (no subsidy).

- μ Dividend return rate for equities in taxable accounts. Average is about 1.7% for S&P 500.
- ν Heirs' income tax rate to be applied on the tax-deferred portion of the estate. This is not an estate tax but rather the federal income marginal tax rate that heirs would have to pay on inherited tax-deferred accounts.
- ϕ_j Fraction of savings account j that is left to surviving spouse i_s as a beneficiary at the death of individual i_d , the first spouse to pass.
- η Spousal ratio for surplus deposits, which goes from 0 to 1, as the fraction that goes to the $i = 1$ spouse's account. Therefore, a surplus s_n in year n will result in a deposit d in the taxable account of individual i as

$$\begin{aligned} d_{0n} &= (1 - \eta)s_n \\ d_{1n} &= \eta s_n. \end{aligned} \tag{4.23}$$

This choice is such that we can set a value depending on the surviving individual $\eta = i_s$ for $n \geq n_d$, after the passing of i_d . Default value is $(N_i - 1)/2$, i.e., 0.5 for couples and 0 for single individuals. When the beneficiary of the savings accounts is not the other spouse, i.e., when $\phi_j \neq 1, \forall j$, it is recommended that η be set to i_d so that all surplus gets deposited to i_d 's accounts, thus avoiding loopholes when optimizing for the final bequest.

- $\mathcal{M}$ This large constant is used in the so-called big- $\mathcal{M}$ method to implement binary constraints. As this is mainly used around MAGI, a value of about 10^8 should be adequate for most cases. We also use $\mathcal{M}$ to represent the upper bounds of top tax brackets.

4.4 Intermediate variables

We use intermediate variables for conciseness or clarity, but they are ultimately replaced in the final formulation. Intermediate variables are represented in roman uppercase letters, or in double stroke uppercase letters.

G_n Taxable ordinary income in year n . Proceeds from the liquidation of fixed assets taxable as ordinary income, the sum of wages, other ordinary income, pensions, taxable Social Security benefits, all withdrawals from tax-deferred accounts, including Roth conversions, and gains from securities (i.e., all gains except those from the $k = 0$ equities, which are taxed as capital gains) in the $(j = 0)$ taxable account, including contributions κ , minus the standard deduction e_n ,

$$\begin{aligned}
G_n = & \mathcal{A}_n^x + \sum_i [\omega_{in} + v_{in} + \nu_{in} + R_n \bar{\zeta}_{in} + \bar{\pi}_{in}] - e_n \\
& + \sum_i [w_{i1n} + x_{in}] \\
& + \sum_{i,k \neq 0} [(b_{i0n} - w_{i0n} + d_{in} + 0.5\kappa_{i0n})\alpha_{i0kn}\tau_{kn}]
\end{aligned} \tag{4.24}$$

Social Security is indexed for inflation. The taxable portion $R_n \sum_i \bar{\zeta}_{in}$ is determined by the self-consistent loop; see the Social Security taxability paragraph in the Constraints section. Pensions can optionally be indexed for inflation. We exclude $k = 0$ to capture only non-equity gains in taxable accounts. These gains are all taxed as ordinary income. Here, we assume that withdrawals and deposits in the taxable account are taking place at the beginning of the year, while contributions, if any, are taking place in mid-year.

G_n was also already defined in Eq. (4.1) as

$$G_n = \sum_{t=0}^{N_t-1} f_{tn},$$

and equating both equations links these variables together.

Q_n Qualified dividends and long-term capital gains obtained in year n . They only involve dividends occurring in the taxable savings accounts $(j = 0)$ that were obtained from equities $(k = 0)$, or sales of stocks due to withdrawals from taxable savings accounts. For simplicity, we assume that all equity sales only generate long-term capital gains and that all dividends are qualified, resulting in

$$Q_n = \sum_i \alpha_{i00n} [(b_{i0n} - w_{i0n} + d_{in} + 0.5\kappa_{i0n})\mu + w_{i0n} \max(0, \tau_{0(n-1)} - \mu)].$$

(4.25)

A formulation where only a fraction of dividends are qualified can easily be implemented with the addition of another parameter. Notice that we are using return rates from the previous year. The first terms on the right-hand side represent dividends generated by equities ($k = 0$) in the ($j = 0$) taxable savings account plus half the yearly contributions. The second term accounts for withdrawals w of equities assumed to have been purchased a year ago. Capital gains are calculated as price appreciation only (total return minus dividend rate) to avoid double taxation of dividends, which are already included in the first term. It does not account for losses, but a market drop would most likely result in stock purchase rather than sale. For withdrawals, we make the assumption of selling the most recent stocks which would not be accurate in situations where the taxable savings account is being depleted slowly. An implementation keeping track of stock purchases and sales is beyond the scope of providing a guide for retirement decisions.

When an initial cost basis K_{i0} is provided for individual i via `setCostBasis()`, the capital-gain coefficient on withdrawals is generalized to a tracked gain fraction ψ_{in} , giving

$$Q_n = \sum_i \alpha_{i00n} [(b_{i0n} - w_{i0n} + d_{in} + \frac{1}{2}\kappa_{i0n})\mu + \psi_{in} w_{i0n}] , \quad (4.26)$$

where

$$\psi_{in} = \max\left(0, 1 - \frac{K_{in}}{b_{i0n}}\right) \quad (4.27)$$

is the fraction of the taxable account balance that represents unrealized gain. Equation (4.25) is recovered when $\psi_{in} = \max(0, \tau_{0(n-1)} - \mu)$, the fallback used in the absence of a known basis. The basis evolves between self-consistent iterations via the pro-rata (average-cost) rule

$$K_{i,n+1} = K_{in} \left(1 - \frac{w_{i0n}}{b_{i0n}}\right) + \kappa_{i0n} + d_{in}, \quad (4.28)$$

initialized at K_{i0} from `setCostBasis()`. Each withdrawal reduces the total basis in proportion to the account fraction liquidated; fixed contributions κ_{i0n} and surplus deposits d_{in} are new purchases added at full cost (zero embedded gain).

$\mathbb{G}_n$ Modified adjusted gross income (MAGI) for year n , on the adjusted-gross-income (AGI) basis. This is the canonical tax MAGI (AGI plus tax-exempt interest) used for IRMAA, the Net Investment Income Tax, and the OBBBA 65+ senior-deduction phaseout. Ignoring non-taxable municipal interest,

$$\mathbb{G}_n = G_n + Q_n + e_n, \quad (4.29)$$

where $G_n + e_n$ already contains *only* the taxable Social Security term $R_n \sum_i \bar{\zeta}_{in}$ (the non-taxable remainder is not part of AGI).

$\mathbb{G}_n^{\text{aca}}$ Full-Social-Security MAGI variant. The Affordable Care Act premium credit (IRC §36B) adds the non-taxable portion of Social Security back to AGI, and the Social Security provisional-income formula likewise uses the full benefit:

$$\mathbb{G}_n^{\text{aca}} = \mathbb{G}_n + (1 - R_n) \sum_i \bar{\zeta}_{in}. \quad (4.30)$$

This approach ignores additional IRS rules around tax-free interests which are insignificant in most cases.

Π_n Provisional income for Social Security taxability in year n (IRS Publication 915): $\Pi_n = \mathbb{G}_n^{\text{aca}} - \frac{1}{2} \sum_i \bar{\zeta}_{in}$. Used to determine the taxable portion of Social Security; see Eq. (5.46) and the Social Security taxability paragraph in the Constraints section.

R_n Effective fraction of Social Security benefits subject to income tax in year n , bounded in $[0, 0.85]$. It is computed by the self-consistent loop from the IRS provisional income formula (IRS Publication 915) and enters the LP as a fixed parameter each iteration. $R_n = \mathcal{S}_n / \sum_i \bar{\zeta}_{in}$; see the Social Security taxability paragraph in the Constraints section.

$\mathcal{S}_n$ Amount of Social Security benefits subject to federal income tax in year n . $\mathcal{S}_n$ is computed from the IRS formula in Eqs. (5.47)–(5.50), and $R_n = \mathcal{S}_n / \sum_i \bar{\zeta}_{in}$.

$\mathbb{I}_n$ Interest and dividend income from the taxable account, plus rent and trust income from the `netinv` column. Used as the base for NIIT computation (together with Q_n).

$$\mathbb{I}_n = \max \left(0, \sum_{i,k \neq 0} [(b_{i0n} - w_{i0n} + d_{in} + 0.5\kappa_{i0n})\alpha_{i0kn}\tau_{kn}] \right) + \sum_i \nu_{in}. \quad (4.31)$$

Only positive returns are taxable; the interest and dividend component is clamped to zero before adding ν_{in} .

P_n Amount of 10% early withdrawal penalty in year n . The penalty applies only to tax-deferred ($j = 1$) withdrawals before the IRS threshold of age $59\frac{1}{2}$:

$$P_n = 0.10 \sum_i (1 - \mathcal{H}(n - n_{i,59\frac{1}{2}})) w_{i1n}. \quad (4.32)$$

Here, $\mathcal{H}(n - n_{i,59\frac{1}{2}})$ is a Heaviside step function which is 0 or 1, depending on the sign of its argument:

$$\mathcal{H}(x) := \begin{cases} 0 & x < 0 \\ 1 & x \geq 0. \end{cases} \quad (4.33)$$

The parameter $n_{i,59\frac{1}{2}}$ is the year index when individual i turns $59\frac{1}{2}$, or 0 if the individual is already past $59\frac{1}{2}$ at the beginning of the plan. Because the model operates at yearly granularity:

$$n_{i,59\frac{1}{2}} = \max(0, 59 - Y_0 + y_{\text{obs},i} + \mathbf{1}[m_{\text{obs},i} > 6]), \quad (4.34)$$

where Y_0 is the current calendar year, $y_{\text{obs},i}$ is the birth year of individual i , and $m_{\text{obs},i} \in \{1, \dots, 12\}$ is the birth month. The indicator $\mathbf{1}[m > 6]$ adds one year for individuals born in July–December, whose $59\frac{1}{2}$ birthday falls in January–June of the following year. Tax-free ($j = 2$) Roth withdrawals are not subject to this penalty because the 5-year maturation constraints already restrict withdrawals to penalty-free amounts; see the maturation paragraph below.

T_n Amount of income tax paid on taxable ordinary income G_n in year n . This is the taxes paid on ordinary income expressed as the sum of the amounts paid in each tax bracket as

$$T_n = \sum_{t=0}^{N_t-1} f_{tn} \theta_{tn}^x. \quad (4.35)$$

Notice how f_{tn} also defines G_n in Eq. (4.1), and that optimal values of f_{tn} have to minimize T_n regardless of whether the bequest or the desired net spending is being maximized.

U_n Amount of income tax paid on long-term capital gains (LTCG) and qualified dividends in year n . The LTCG tax is computed exactly by the LP bracket variables q_{pn} :

$$U_n = 0.15 q_{1n} + 0.20 q_{2n}. \quad (4.36)$$

We assume that qualified dividends and long-term capital gains are taxed at the same preferential rate, which is the case for most situations.

J_n Net investment income tax (NIIT) paid in year n . The NIIT is 3.8% and applies when modified adjusted gross income $\mathbb{G}_n$ exceeds the statutory threshold $\mathbb{G}^{\text{NIIT}}$ (formally defined in the Parameters section; not indexed for inflation). We compute

$$J_n = 0.038 \max \left(0, \min \left(\mathbb{G}_n - \mathbb{G}^{\text{NIIT}}, \mathbb{I}_n + Q_n \right) \right), \quad (4.37)$$

where $\mathbb{G}^{\text{NIIT}}$ is \$200k for single filers and \$250k for married filing jointly, with filing status changing after the passing of one spouse ($n \geq n_d$). In the default SC-loop mode J_n is a derived quantity, recomputed after each LP solve from the current $\mathbb{G}_n$ and held fixed during the solve. When **withNIIT="optimize"** is selected, J_n and $\mathbb{G}_n$ become LP decision variables, and a binary variable z_n^j exactly determines whether the NIIT threshold is exceeded (see the Net Investment Income Tax subsection of the Constraints chapter).

K_{in} Average-cost basis of the taxable account for individual i at the start of year n . Initialized from the user-supplied value K_{i0} via **setCostBasis()** and updated each self-consistent iteration by Eq. (4.28). Not used when no cost basis is provided.

5. Formulation with imposed asset allocation ratios

We first present the case where the sums of assets in each savings account b_{ijn} are known and for which we assume prescribed asset allocation ratios. The amount in each asset class k for b_{ijkn} is simply obtained from $\alpha_{ijkn}b_{ijn}$ in this case. This formulation assumes that the accounts are always balanced. This is a reasonable assumption given the auto-balancing feature offered by many financial service providers and robo-advisers.

The benefit of this approach is that it has fewer variables and that only the sums of all asset classes in each savings account need to be considered. The rate of return of the account is then simply the product of the account balance with the sum of the rates of return weighted according to the desired allocation ratio. This approach allows us to eliminate k by summing over it and rewrite equations such as

$$\sum_k b_{ijk(n+1)} = \sum_k b_{ijkn}(1 + \tau_{kn}) + \dots, \quad (5.1)$$

for the annual evolution of account balances from year n to year $n + 1$ as the simpler expression

$$\begin{aligned} b_{ij(n+1)} &= b_{ijn} \sum_k \alpha_{ijkn}(1 + \tau_{kn}) + \dots, \\ &= b_{ijn}(1 + \mathcal{T}_{ijn}) + \dots, \end{aligned} \quad (5.2)$$

where

$$\mathcal{T}_{ijn} := \sum_k \alpha_{ijkn} \tau_{kn}. \quad (5.3)$$

It follows that the allocation ratios are normalized to unity, i.e.,

$$\sum_k \alpha_{ijkn} = 1. \quad (5.4)$$

In this formulation where the α_{ijkn} are prescribed, we will use $\mathcal{T}_{ijn}$ to add the market returns to the savings balances.

5.1 Constraints

5.1.1 Required minimum distributions (RMDs)

Withdrawals from the ($j = 1$) tax-deferred savings accounts must be greater than or equal to the required minimum distributions, and therefore,

$$w_{i1n} - \rho_{in} b_{i1n} \geq 0. \quad (5.5)$$

As b_{ijn} are the balances at the beginning of year n , they are also the balances at December 31 of the previous year, which is the amount on which the IRS bases the RMDs. Eq. (5.5) has to hold for each year n and each individual i , and therefore, there are $N_i \times N_n$ such equations (although trivial when $\rho_{in} = 0$). These constraints avoid paying up to 25% penalty on amounts not withdrawn when RMDs are required. Note that aggregate rules need to be considered separately as this approach only considers the sum of assets in a class with similar tax treatment (e.g., IRA and 401k).

5.1.2 Income tax brackets

Taxable ordinary income is divided into tax brackets as defined in Eq. (4.1), and therefore

$$0 \leq f_{tn} \leq \bar{\Delta}_{tn}, \quad (5.6)$$

where $\bar{\Delta}_{tn}$ is the inflation-adjusted income tax bracket width.

5.1.3 Standard exemption

The standard exemption is constrained by

$$0 \leq e_n \leq \bar{\sigma}_n. \quad (5.7)$$

Variable e_n is required for accounting for years when the taxable ordinary income is smaller than the standard exemption.

5.1.4 Withdrawal limits

We introduce another set of constraints that might look unnecessary, but can help convergence, and prevent overdrafts during the year of passing of spouse i_d . As withdrawals and conversions are at the beginning of the year we impose that

$$w_{ijn} + \delta_{j,1}x_{in} \leq b_{ijn}, \quad (5.8)$$

which just states that account balances need to be at least as large as withdrawals and possible Roth conversions.

5.1.5 Withdrawal ordering

By default, the optimizer is free to choose the order in which accounts are drawn down. When `withdrawalOrder="taxable_first"` is selected, the conventional ordering — taxable first, then tax-deferred, then Roth — is enforced instead, modeling the naive strategy a hand-managed plan would follow; it serves as a baseline policy when quantifying the value added by the optimization. Two household-level binary gates per year, z_{0n}^o and z_{1n}^o , indicate that the taxable and the tax-deferred balances, respectively, are exhausted at the end of year n . Withdrawals beyond what the gates allow are blocked through

$$\begin{aligned} w_{i1n} - \rho_{in} b_{i1n} &\leq \mathcal{M}\gamma_n z_{0n}^o, \\ w_{i2n} &\leq \mathcal{M}\gamma_n z_{1n}^o, \end{aligned} \quad (5.9)$$

so that, while the taxable gate is closed ($z_{0n}^o = 0$), the tax-deferred withdrawal is pinned to exactly the required minimum distribution by Eq. (5.5), and no Roth withdrawal can occur while the tax-deferred gate is closed. A gate may open only when the corresponding household balance is exhausted at the end of the year,

$$\begin{aligned} \sum_i b_{i0(n+1)} &\leq \mathcal{M}\gamma_n (1 - z_{0n}^o), \\ \sum_i b_{i1(n+1)} &\leq \mathcal{M}\gamma_n (1 - z_{1n}^o), \end{aligned} \quad (5.10)$$

and the full ordering follows from the monotonicity condition

$$z_{1n}^o \leq z_{0n}^o. \quad (5.11)$$

HSA withdrawals (qualified medical expenses) are not gated. Because surplus deposits land in the taxable account at year end, a year cannot both deposit a surplus and claim the taxable-exhausted gate, which closes the loophole of draining an account and immediately redepositing.

5.1.6 Posthumous account activities

For cases with spouses, no withdrawal, Roth conversion, or deposit should occur in the accounts of the passed individual i_d :

$$\begin{aligned} w_{i_d j n} &= 0, \\ d_{i_d n} &= 0, \\ x_{i_d n} &= 0, \\ &\quad \forall j \in \{0, \dots, N_j - 1\} \\ &\quad \forall n \in \{n_d, \dots, N_n - 1\}. \end{aligned} \tag{5.12}$$

5.1.7 Roth conversions

Roth conversions x_{in} are forced to zero in the last two years of each individual's horizon (see Section 2.7). Otherwise, Roth conversions cannot be larger than the balance at the beginning of the year in the account:

$$x_{in} \leq b_{i1n}. \tag{5.13}$$

This constraint, however, is naturally satisfied when $b_{ijn} \geq 0$ non-negativity bounds are enforced. Additional maximum Roth conversion constraints x_{max} can be imposed by the user. For a single individual, the previous equation becomes

$$x_{in} \leq \min(b_{i1n}, x_{max}). \tag{5.14}$$

For couples, the cap applies to each individual's conversions:

$$x_{in} \leq x_{max}, \quad i \in \{0, 1\}. \tag{5.15}$$

As these equations involve a variable and a parameter in the min function, they are coded as separate constraints.

5.1.8 Minimum taxable balance

To maintain liquidity throughout the plan, the user may impose a lower bound $\beta_i^{\min}$ (in today's dollars) on the taxable account balance of each individual. The inflation-adjusted floor is enforced from year 1 through the end of the individual's life horizon:

$$b_{i0n} \geq \beta_i^{\min} \gamma_n, \quad n \in \{1, \dots, \text{horizon}_i - 1\}. \tag{5.16}$$

This is controlled by the `minTaxableBalance` option. The constraint is omitted when `minTaxableBalance` is not specified.

5.1.9 Initial balances

The initial balances β_{ij} are one of the main inputs of the model. The initial savings account balances are imposed through the constraints

$$b_{ij0} = \beta_{ij}. \quad (5.17)$$

At this point, we assume that all accounts are balanced according to the desired allocation ratios α_{ijk0} .

5.1.10 Roth 5-year maturation and conversion ladder

Two IRS rules govern penalty-free access to Roth funds. (1) Roth *contributions* can always be withdrawn tax- and penalty-free. (2) Roth *conversions* can be withdrawn penalty-free only after a 5-year holding period, with each conversion carrying its own clock; *earnings* additionally require age 59½. Because of rule (2), any Roth withdrawal allowed by the maturation constraint is already penalty-free, so no 10% early-withdrawal penalty is assessed on w_{i2n} .

The maturation constraint (row group I_8 in the matrix appendix) enforces this rule:

$$b_{i2n} - w_{i2n} \geq \sum_{k=1}^5 \Gamma_k x_{i,n-k} + \sum_{k=1}^5 (\Gamma_k - 1) \kappa_{i2,n-k}, \quad \Gamma_k = \prod_{\ell=1}^k \max(1, 1 + \mathcal{T}_{i2,n-\ell}), \quad (5.18)$$

forcing immature conversions (those made within the last 5 years) and recent contribution gains to remain in the account. Historical amounts ($n - k < 0$) use a conservative 10%/yr assumption.

This constraint enables the *Roth conversion ladder*, a strategy popular in the FIRE (Financial Independence, Retire Early) community. An early retiree converts tax-deferred funds in years $n, n+1, \dots$ and, 5 years later, withdraws the corresponding Roth amounts penalty-free. The optimizer discovers this strategy automatically once the Roth penalty is removed and the AMO constraint is relaxed for pre-59½ years (see the AMO discussion below).

5.1.11 Cash-flow surplus

When both spouses are alive, surplus s_n gets deposited in the taxable accounts according to variable η as described in Eq. (4.23),

$$d_{in} = [\delta_{i,0}(1 - \eta) + \delta_{i,1}\eta]s_n. \quad (5.19)$$

Otherwise, for $n \geq n_d$, variable η gets redefined as $\eta = \delta_{1,i_s}$. Surplus can be caused by large influx of money coming from big-ticket items, compulsory RMDs, or the disposition of fixed assets. The `noLateSurplus` option pins surplus to zero in the final two plan years. This is useful because during market downturns, large sums can otherwise be recycled through the taxable account in those years with little tax or growth consequence, since the long-term compounding benefit of such deposits vanishes near the end of the plan.

5.1.12 Account balances

Contributions are assumed to be made at half-year to better represent periodic contributions made throughout the year. As we already mentioned, the account balance at the end of a year is the same as the balance at the beginning of the following year. Changes include contributions κ , distributions and withdrawals w , conversions x , surplus deposits d , and growth τ on the account through the year. For each spouse i , we track each savings account j separately, and tax-deferred accounts are coupled with the corresponding tax-free account through Roth conversions.

The timing of Roth conversions, withdrawals, and deposits brings additional coupling between these variables, and is worth a detailed discussion. First, the financial aspects, and then the algorithmic ones. For the former, some financial advisors would recommend making Roth conversions at the beginning of the year, while making withdrawals at the end. Obviously, financial simulators would always yield higher numbers when using this scenario, as the money needed to pay the regular bills stayed in the bank until the end of the year. More realistically, however, it would be more accurate to assume withdrawals at mid-year, to better represent evenly distributed withdrawals. So, financially, conversions at the beginning of the year, and withdrawals at mid-year make good sense. Conversions are also typically best when timed with market downturns, which are obviously not always at the beginning of the year.

Now, let's look at the optimization side of these transactions. During years of positive returns, a direct withdrawal from the tax-deferred account at mid-year will always be unfavorable when compared to a Roth conversion at the beginning of the year, followed by a tax-free withdrawal later in the same year. This is because the second scenario involves gains which are tax-free over the half-year, while the first one does not. Moving account withdrawals at the beginning of the year, and the conversions in mid-year can only solve part of this artificial bias under specific conditions.

Moving all transactions at the beginning of the year can also cause undesired

transactions. To solve these spurious scenarios, it would be desirable to make the following at-most-one (AMO) exclusions between surplus s_n , withdrawals w_{ijn} , and conversions x_{in} :

$$\left(\sum_i w_{i0n} + \sum_i w_{i2n} \right) \text{ AMO } s_n, \\ \sum_i x_{in} \text{ AMO } \sum_i w_{i2n},$$

for $j \neq 1$, i.e., for all withdrawals except those from tax-deferred accounts. The AMO relation can also be represented by a logical NAND operator. The first exclusion prevents surplus deposits when withdrawals from taxable or tax-free accounts occur from either spouse, while the second exclusion prevents simultaneous Roth conversions and tax-free withdrawals from either spouse. The natural way to impose these exclusions is with binary variables and big- $\mathcal{M}$ constraints, one pair per exclusion per year. Owl did exactly that, and no longer does. The reason is worth setting out, because it says something about the objective rather than about the constraints.

Why the binaries were removed Measured across every shipped example case under both solvers, the constrained optimum equals that of the relaxation: the exclusions never change what the plan is worth. They are symmetry breakers, selecting the legible member of a family of equally valued solutions. They are not free, however. Owl’s objective distinguishes the three account types *only* through their tax treatment, so a household with a zero marginal rate on both ordinary income and capital gains gains nothing from where a dollar sits. The objective then goes flat across a combinatorial set of plans, and branch-and-bound loses the ability to prune: it holds the optimal value at the root node and spends the rest of its time proving no tie beats it. The effect is severe. A single retiree with about \$119,000 of savings, no pension, and Social Security of \$200 per month solves in 0.05 seconds without the binaries and took 577 seconds with them.

Two details of that pathology are worth recording. First, a loose $\mathcal{M}$ is not the cause: tightening it by two orders of magnitude changes the solve time by a quarter, because the bound was already exact and the failure is discrimination rather than tightness. Second, zero federal tax is necessary but not sufficient. The same household in a state with no income tax solves in 0.06 seconds; it is the state deduction and exemption variables, free to move when there is no tax to pay, that widen the flat face far enough to matter. These exclusions were also the only binary family a default plan carried, so removing them turns the default problem into a pure linear program.

Restoring the exclusions after the solve Both properties are re-established on the solved vector instead. For the Roth exclusion, a year $n \geq n_{i,59\frac{1}{2}}$ carrying both a conversion and a tax-free withdrawal is rewritten with $m = \min(x_{in}, w_{i2n})$ as

$$x_{in} \rightarrow x_{in} - m, \quad w_{i2n} \rightarrow w_{i2n} - m, \quad w_{i1n} \rightarrow w_{i1n} + m. \quad (5.20)$$

Every affected row is invariant rather than merely slack. The two balance carry-overs cancel, the second of them provided $\bar{x} = 1$; taxable income is unchanged because a conversion and a tax-deferred withdrawal are both fully taxable; net cash is unchanged once the ten-percent early-withdrawal penalty is absent, which is what confines the substitution to $n \geq n_{i,59\frac{1}{2}}$; and the tax-deferred withdrawal limit $b_{i1n} - w_{i1n} - x_{in} \geq 0$ is invariant term by term. The RMD floor, the five-year Roth retainer and the remaining withdrawal limits only loosen. The substitution is therefore exact and needs no second solve.

Confining it to $n \geq n_{i,59\frac{1}{2}}$ is not merely a technical requirement: before an individual reaches $59\frac{1}{2}$ it is legitimate to withdraw mature five-year-old conversions while initiating new ladder rungs, so removing that overlap would dismantle the Roth conversion ladder. The pre- $59\frac{1}{2}$ years are left exactly as solved.

For the surplus exclusion, note first that the surplus is deposited into the same taxable account a withdrawal may have just drawn from, so the gross figures describe a movement that does not occur. Reporting the two net of one another is an identity for the cash flow balance, since the surplus appears on its left-hand side and the deposit within the withdrawals on its right, and cancelling them removes the same amount from both. Balances, taxes, spending and the bequest are untouched. Years whose capital gains reach a taxed bracket are left gross, so that a reported withdrawal always accounts for the capital-gains tax reported beside it. Whatever overlap survives is re-examined by a churn-minimizing re-solve of the same program with spending, the terminal balances and the conversion schedule pinned, which cannot change any reported objective and for which the incumbent is itself a feasible point.

What is not recovered Three situations admit no exact substitution and are reported rather than removed: a non-zero opportunity cost on conversions, for which $\bar{x} \neq 1$ breaks the balance invariance; the `taxable\_first` withdrawal ordering, whose gate pins tax-deferred withdrawals to the RMD and so forbids the substitution's larger withdrawal; and a couple in which one spouse converts while the other withdraws, since the original exclusion was a household-level row while the substitution is per individual. In each case the overlap is left in the reported plan, which costs nothing, rather than paid for out of the objective. Two vanishingly small preferences

in the objective—for leaving a dollar in an account that is never taxed again, and for not withdrawing money only to bank it—keep the solvers from selecting arbitrarily among the remaining ties.

Another approach could be to perform Roth conversions at mid-year, while withdrawals could be made at the beginning of the year, and surplus deposits, if needed due to RMDs or receiving large sums of money, could be made at the end of the year. Let’s formulate this approach in more detail and investigate for potential problems. Timing controls which terms get multiplied by the rate of return $(1 + \mathcal{T}_{ijn})$. Therefore, our current choice would yield

$$\begin{aligned} b_{ij(n+1)} = & [b_{ijn} - w_{ijn} + 0.5\kappa_{ijn}](1 + \mathcal{T}_{ijn}) + [\delta_{j,2} - \delta_{j,1}]x_{in}(1 + \mathcal{T}_{ijn}/2) \\ & + \delta_{j,0}d_{in} + 0.5\kappa_{ijn}, \end{aligned} \quad (5.21)$$

where we use discrete Kronecker δ functions for selecting the specific accounts involved in Roth conversions. These conversions are made such that asset allocation ratios in the sending and receiving accounts are unchanged.

Bringing all variables to the left-hand side, this gets rewritten as

$$\begin{aligned} & b_{ij(n+1)} - (b_{ijn} - w_{ijn})(1 + \mathcal{T}_{ijn}) \\ & - [\delta_{j,2} - \delta_{j,1}]x_{in}(1 + \mathcal{T}_{ijn}/2) - \delta_{j,0}d_{in} = \kappa_{ijn}(1 + \mathcal{T}_{ijn}/2). \end{aligned} \quad (5.22)$$

When $j = 0$, this equation introduces a path to shelter negative returns by performing an over-withdrawal from the taxable account at the beginning of the year followed by a deposit in the same account at the end of the year. This can be removed by making these events exclusive. Owl handles the same round-trip after the solve instead, by reporting the withdrawal net of the deposit that returns to the account (Eq. (5.20) and the discussion around it), which avoids the binary variable and the branch-and-bound cost that comes with it.

A much simpler approach, while not so natural, is to move all transactions to be synchronous at the beginning or at the end of the year thus avoiding undesirable movements of funds. If we select the beginning of the year, except for contributions, this leads to

$$b_{ij(n+1)} - [b_{ijn} + \delta_{j,0}d_{in} - w_{ijn} + (\delta_{j,2} - \delta_{j,1})x_{in}](1 + \mathcal{T}_{ijn}) = \kappa_{ijn}(1 + \mathcal{T}_{ijn}/2).$$

This is the current approach used in Owl, coupled with binary variables excluding simultaneous overwithdrawals and deposits, and simultaneous Roth conversions and withdrawals from tax-free accounts.

5.1.13 Net spending

For calculating the net spending g_n , we consider the cash flow of all withdrawals, wages, other ordinary income, Social Security and pension benefits, proceeds from liquidation of fixed assets, and big-ticket items. Then we subtract potential surplus s_n and all taxes, penalties, debts, Medicare premiums, and ACA premiums paid:

$$g_n = \sum_i [\omega_{in} + v_{in} + \nu_{in} + \bar{\zeta}_{in} + \bar{\pi}_{in}] + \sum_{i,j} w_{ijn} + \sum_i \Lambda_{in}^{\pm} + \sum_{*=x,c,f} \mathcal{A}_n^* - s_n - P_n - T_n - U_n - J_n - m_n - m_n^{\text{aca}} - \mathcal{D}_n. \quad (5.23)$$

Here m_n^{aca} is the ACA net premium cost. It is a decision variable when `withACA="optimize"`, or a fixed SC-loop parameter when `withACA="loop"`. Notice how big-ticket items $\Lambda^{\pm}$ contribute directly to the cash flow. Replacing intermediate variables and bringing all variables to the left-hand side, we get

$$\begin{aligned} g_n - \sum_{i,j} w_{ijn} + 0.1 \sum_i (1 - \mathcal{H}(n - n_{i,59\frac{1}{2}})) w_{i1n} \\ + m_n + m_n^{\text{aca}} + s_n + \sum_{t=0}^{N_t-1} f_{tn} \theta_{tn}^x + U_n = \sum_{*=x,c,f} \mathcal{A}_n^* + \sum_i \Lambda_{in}^{\pm} \\ + \sum_i [\omega_{in} + v_{in} + \nu_{in} + \bar{\zeta}_{in} + \bar{\pi}_{in}] - \mathcal{D}_n - J_n. \end{aligned} \quad (5.24)$$

When `withACA="loop"`, m_n^{aca} is not an LP variable; the ACA cost is computed in the SC loop and subtracted on the RHS (omitted from the equation above for clarity). Notice that we do not consider market losses or tax-loss harvesting, and we do not track individual stock purchases over the years. We also treat J_n as a parameter as it is computed through a self-consistent loop.

We want the net spending to be predictable and smooth. For that purpose, we use

$$g_n / \bar{\xi}_n = g_0 / \bar{\xi}_0, \quad (5.25)$$

where the net spending is adjusted for inflation and where we use the time series of parameter ξ_n , allowing for additional adjustments to the overall desired spending. Note that $\bar{\xi}_0 = \xi_0$ as $\gamma_0 = 1$. This profile is used to lower the desired net spending amount by a reduction factor χ after the passing of one spouse and/or to allow for

more realistic spending profiles, such as the *smile* profile described above. Eq. (5.25) can be rewritten as

$$g_n \bar{\xi}_0 - g_0 \bar{\xi}_n = 0, \quad (5.26)$$

for the constraints to be enforced. Once g_0 is determined, the whole time series of net spending is determined. When using slack variable λ , the spending profile is then implemented as inequality constraints

$$\begin{aligned} g_n \bar{\xi}_0 - g_0(1 - \lambda) \bar{\xi}_n &\geq 0, \\ -g_n \bar{\xi}_0 + g_0(1 + \lambda) \bar{\xi}_n &\geq 0. \end{aligned} \quad (5.27)$$

In this case, the objective function will need to consider the sum over all years when optimizing net spending.

5.1.14 LTCG bracket constraints

The LTCG tax $U_n = 0.15 q_{1n} + 0.20 q_{2n}$ is determined by three LP bracket allocation variables q_{pn} ($p \in \{0, 1, 2\}$). The partition constraint

$$q_{0n} + q_{1n} + q_{2n} \geq Q_n \quad (5.28)$$

ensures the allocations cover the total LTCG Q_n . The bracket upper-bound constraints prevent overflow into higher-rate brackets. Because LTCG is stacked on top of ordinary taxable income $G_n = \sum_{t=0}^{N_t-1} f_{tn}$, the room available in each bracket depends on how much of the threshold is already occupied by ordinary income. The constraints can be expressed as

$$q_{0n} \leq \max(0, \bar{\mathcal{B}}_n^{15} - G_n), \quad (5.29)$$

$$q_{0n} + q_{1n} \leq \max(0, \bar{\mathcal{B}}_n^{20} - G_n), \quad (5.30)$$

where $\bar{\mathcal{B}}_n^{15}$ and $\bar{\mathcal{B}}_n^{20}$ are the inflation-adjusted total taxable income thresholds above which the 15% and 20% LTCG rates apply. In these constraints, G_n is taken from the solution of the previous SC loop iteration (initialized to zero on the first pass). These constraints converge together with the SC loop. Since the cost is convex ($0\% < 15\% < 20\%$), no binary variables are needed: the LP naturally allocates gains to the cheapest bracket first.

When the option `withLTCG="optimize"` is selected, the bracket thresholds are no longer fixed from a previous iteration: G_n itself becomes a continuous LP variable, and two binary variables $z_{0n}^l, z_{1n}^l \in \{0, 1\}$ (per year n) indicate whether room exists

below the 15% and 20% thresholds, respectively ($z^l = 1$ means room exists, G_n is below the threshold). The big- $\mathcal{M}$ link constraints are

$$\bar{\mathcal{B}}_n^{15} \leq G_n + z_{0n}^l \mathcal{M}_n, \quad (5.31)$$

$$G_n + z_{0n}^l \mathcal{M}_n \leq \bar{\mathcal{B}}_n^{15} + \mathcal{M}_n, \quad (5.32)$$

and analogously for z_{1n}^l at threshold $\bar{\mathcal{B}}_n^{20}$, with $\mathcal{M}_n = 3\bar{\mathcal{B}}_n^{20}$ a safe upper bound. A monotonicity constraint

$$z_{0n}^l \leq z_{1n}^l \quad (5.33)$$

ensures consistency (if G_n is below the 15% threshold it is also below the 20% threshold). The bracket room upper bounds then become linear:

$$q_{0n} \leq z_{0n}^l \mathcal{M}_n, \quad (5.34)$$

$$q_{0n} + q_{1n} \leq z_{1n}^l \mathcal{M}_n. \quad (5.35)$$

Because the cost is still convex, the optimizer fills the cheapest bracket first, so no additional binary variables are needed for bracket ordering; only the bracket room bounds require binaries.

5.1.15 Net investment income tax

In the default SC-loop mode, the NIIT J_n is derived from MAGI and investment income after each solver pass and held fixed for the next LP solve. When the option `withNIIT="optimize"` is selected, MAGI $\mathbb{G}_n$ becomes a continuous LP variable `magin`, a binary variable $z_n^j \in \{0, 1\}$ determines whether MAGI exceeds the statutory NIIT threshold $\mathbb{G}^{\text{NIIT}}$, and J_n becomes a continuous LP variable.

Threshold indicator

The big- $\mathcal{M}$ constraints enforcing $z_n^j = 1$ iff $\mathbb{G}_n \geq \mathbb{G}^{\text{NIIT}}$ are

$$J_n \leq z_n^j \mathcal{M}_n^j, \quad (5.36)$$

$$\mathbb{G}_n \leq \mathbb{G}^{\text{NIIT}} + z_n^j \mathcal{M}_n^j, \quad (5.37)$$

with $\mathcal{M}_n^j = 3\bar{\mathcal{B}}_n^{20}$. Constraint (5.36) forces $J_n = 0$ when $z_n^j = 0$; Constraint (5.37) forces $\mathbb{G}_n \leq \mathbb{G}^{\text{NIIT}}$ when $z_n^j = 0$, so the optimizer assigns $z_n^j = 0$ only when the threshold is not exceeded.

NII cap via a surplus variable

The IRS formula is $J_n = 0.038 \min(\mathbb{G}_n - \mathbb{G}^{\text{NIIT}}, \mathbb{I}_n + Q_n)$, where $\mathbb{I}_n + Q_n$ is net investment income (NII). High ordinary income (e.g. large RMDs or Roth conversions) can push $\mathbb{G}_n - \mathbb{G}^{\text{NIIT}}$ above NII, in which case the NIIT base is capped at NII. This $\min(\cdot, \cdot)$ is modeled *without a second binary* by introducing a continuous surplus variable $\delta_n^j \geq 0$. Since the optimizer minimizes taxes and J_n is a cost, it is naturally driven to its lower bound; making that lower bound depend on δ_n^j means the optimizer simultaneously maximizes δ_n^j , driving it to $\max(0, \mathbb{G}_n - \mathbb{G}^{\text{NIIT}} - \mathbb{I}_n - Q_n)$. The two constraints are

$$J_n + 0.038 \delta_n^j \geq 0.038 (\mathbb{G}_n - \mathbb{G}^{\text{NIIT}}) - \mathcal{M}_n^j (1 - z_n^j), \quad (5.38)$$

$$\delta_n^j - G_n - e_n + t_n^\sigma \leq \sum_i \bar{\zeta}_{in} - \mathbb{G}^{\text{NIIT}} - \mathbb{I}_n + \mathcal{M}_n^j (1 - z_n^j). \quad (5.39)$$

Constraint (5.38) is the combined MAGI/NII floor: when $z_n^j = 1$ it forces $J_n \geq 0.038 (\mathbb{G}_n - \mathbb{G}^{\text{NIIT}} - \delta_n^j)$. Constraint (5.39) bounds the surplus: $\delta_n^j \leq \mathbb{G}_n - \mathbb{G}^{\text{NIIT}} - \mathbb{I}_n - Q_n$ when $z_n^j = 1$. The Q_n and Social Security terms cancel because, on the AGI basis, $\mathbb{G}_n = G_n + e_n + Q_n$ and $\text{NII}_n = \mathbb{I}_n + Q_n$, so $\mathbb{G}_n - \text{NII}_n = G_n + e_n - \mathbb{I}_n$; thus Constraint (5.39) involves only ordinary income, deduction headroom, and the SC-loop parameter $\mathbb{I}_n$. Together, Constraints (5.38)–(5.39) reproduce $J_n = 0.038 \min(\mathbb{G}_n - \mathbb{G}^{\text{NIIT}}, \mathbb{I}_n + Q_n)$ without any additional binary variables.

MAGI equality

An equality constraint links the NIIT MAGI (AGI basis) to ordinary income, dividends, and LTCG. Because $G_n + e_n$ already carries the taxable Social Security portion $R_n \sum_i \bar{\zeta}_{in}$, no additional Social Security term is added:

$$\mathbb{G}_n = G_n + Q_n + e_n. \quad (5.40)$$

In the LP, Q_n is represented directly by the LTCG bracket allocation variables $q_n^{(0)} + q_n^{(1)} + q_n^{(2)}$, which equal Q_n when the partition lower-bound constraint (5.28) is tight. The raw portfolio capital-gain LP expression must *not* be substituted for Q_n here: it evaluates to zero at the partition minimum and would remove Q_n from MAGI, causing the NIIT to be computed as $0.038 \mathbb{I}_n$ rather than $0.038 (\mathbb{I}_n + Q_n)$.

5.1.16 Taxable ordinary income

We connect the two definitions for G_n stated above in Eqs. (4.1) and (4.24),

$$\begin{aligned}
\sum_{t=0}^{N_t-1} f_{tn} &= \mathcal{A}_n^x + \sum_i [\omega_{in} + v_{in} + \nu_{in} + R_n \bar{\zeta}_{in} + \bar{\pi}_{in}] \\
&\quad + \sum_i [w_{i1n} + x_{in}] \\
&\quad + \sum_{i,k \neq 0} [(b_{i0n} - w_{i0n} + d_{in} + 0.5\kappa_{i0n})\alpha_{i0kn}\tau_{kn}] - e_n,
\end{aligned} \tag{5.41}$$

and rearrange to move variables to the LHS as follows

$$\begin{aligned}
e_n + \sum_{t=0}^{N_t-1} f_{tn} - \sum_i [w_{i1n} + x_{in}] \\
- \sum_{i,k \neq 0} [(b_{i0n} - w_{i0n} + d_{in})\alpha_{i0kn}\tau_{kn}] &= \mathcal{A}_n^x + \sum_i [\omega_{in} + v_{in} + \nu_{in} + R_n \bar{\zeta}_{in} + \bar{\pi}_{in}] \\
&\quad + 0.5 \sum_{i,k \neq 0} \alpha_{i0kn}\tau_{kn}\kappa_{i0n}.
\end{aligned} \tag{5.42}$$

5.1.17 State income tax

When the configured state levies an income tax, state ordinary income is allocated across the N_s state brackets through variables f_{tn}^s , in direct analogy with Eq. (5.6),

$$0 \leq f_{tn}^s \leq \bar{\Delta}_{tn}^s, \tag{5.43}$$

where $\bar{\Delta}_{tn}^s$ are the inflation-adjusted state bracket widths. Variable e_n^s absorbs the state standard deduction through $0 \leq e_n^s \leq \bar{\sigma}_n^s$ and, when the state offers a retirement-income exemption, variable r_n^s , bounded by the inflation-adjusted per-person cap $\bar{c}_n^r$ as $0 \leq r_n^s \leq \bar{c}_n^r$, exempts eligible retirement income. State taxable income is tied to the federal quantities through the equality

$$\sum_{t=0}^{N_s-1} f_{tn}^s + e_n^s + r_n^s - \sum_{t=0}^{N_t-1} f_{tn} - \sum_{p=0}^2 q_{pn} = -E_n^\sigma - E_n^\pi, \tag{5.44}$$

where $E_n^\sigma = R_n \sum_i \bar{\zeta}_{in}$ when the state does not tax Social Security — removing the federally taxable portion embedded in Eq. (5.42), with R_n the self-consistent

taxability ratio — and $E_n^\sigma = 0$ otherwise, and where $E_n^\pi = \min(\sum_i \bar{\pi}_{in}, N_i \bar{c}_n^\pi)$ applies the state pension exemption with per-person cap $\bar{c}_n^\pi$. The retirement-income exemption cannot exceed the tax-deferred income actually received,

$$r_n^s \leq \sum_i w_{in}. \quad (5.45)$$

Finally, the state tax $\sum_t \theta_{tn}^s f_{tn}^s$, with θ_{tn}^s the state marginal rates, is added to the tax terms of the cash-flow constraint Eq. (5.24).

5.1.18 Social Security taxability

Under IRS Publication 915, the fraction of Social Security benefits subject to income tax is determined by the *provisional income* (PI), defined as

$$\Pi_n = \mathbb{G}_n - \frac{1}{2} \sum_i \bar{\zeta}_{in}. \quad (5.46)$$

The IRS piecewise formula gives the taxable SS amount as

$$\mathcal{S}_n = \begin{cases} 0 & \Pi_n < \mathcal{P}^{\text{lo}}, \\ 0.5 (\Pi_n - \mathcal{P}^{\text{lo}}) & \mathcal{P}^{\text{lo}} \leq \Pi_n < \mathcal{P}^{\text{hi}}, \\ \min\left(0.85 \sum_i \bar{\zeta}_{in}, 0.5 (\mathcal{P}^{\text{hi}} - \mathcal{P}^{\text{lo}}) + 0.85 (\Pi_n - \mathcal{P}^{\text{hi}})\right) & \Pi_n \geq \mathcal{P}^{\text{hi}}, \end{cases} \quad (5.47)$$

so that $R_n = \mathcal{S}_n / \sum_i \bar{\zeta}_{in} \in [0, 0.85]$.

Introducing the excess provisional income above each threshold,

$$p_n^{\sigma, \text{lo}} = \max(0, \Pi_n - \mathcal{P}^{\text{lo}}), \quad p_n^{\sigma, \text{hi}} = \max(0, \Pi_n - \mathcal{P}^{\text{hi}}), \quad (5.48)$$

the three cases of Eq. (5.47) collapse into the single expression

$$\mathcal{S}_n = \min\left(0.85 \sum_i \bar{\zeta}_{in}, 0.50 \min\left(\sum_i \bar{\zeta}_{in}, \min(\Delta \mathcal{P}, p_n^{\sigma, \text{lo}})\right) + 0.85 p_n^{\sigma, \text{hi}}\right), \quad (5.49)$$

where $\Delta \mathcal{P} = \mathcal{P}^{\text{hi}} - \mathcal{P}^{\text{lo}}$ is the bracket width (\$12,000 for MFJ; \$9,000 for single filers). The inner $\min(\sum_i \bar{\zeta}_{in}, \cdot)$ guards against the unusual case $\sum_i \bar{\zeta}_{in} < \Delta \mathcal{P}$, in which SS benefits are too small to fill the 50%-bracket entirely. Because the clipped

excess $\min(\Delta\mathcal{P}, p_n^{\sigma,\text{lo}}) \leq \Delta\mathcal{P}$, whenever $\sum_i \bar{\zeta}_{in} \geq \Delta\mathcal{P}$ the inner min is automatically satisfied and Eq. (5.49) simplifies to

$$\mathcal{S}_n = \min\left(0.85 \sum_i \bar{\zeta}_{in}, 0.5 p_n^{\sigma,\text{min}} + 0.85 p_n^{\sigma,\text{hi}}\right), \quad p_n^{\sigma,\text{min}} = \min(\Delta\mathcal{P}, p_n^{\sigma,\text{lo}}). \quad (5.50)$$

This condition holds for virtually all retirees, since combined annual SS benefits nearly always exceed \$12,000 (MFJ) or \$9,000 (single).

Because R_n enters the LP only as a constant multiplier of the fixed series $\bar{\zeta}_{in}$ on the right-hand side of Eq. (5.42), it is treated as a *parameter* rather than a decision variable. It is updated between LP solves in the self-consistent (SC) loop: after each solve, $\mathbb{G}_n$ is computed from the current solution, Π_n is evaluated, and R_n is refreshed via Eq. (5.47). A 30% damping blend $R_n \leftarrow 0.3 R_n^{\text{new}} + 0.7 R_n$ prevents oscillation near the bracket boundaries. To avoid unnecessary LP rebuilds, R_n is only updated when $\max_n |\Delta R_n| > 10^{-3}$. The SC loop terminates when the change in the objective function (net spending or bequest) falls below a prescribed tolerance.

Alternatively, when the solver option `withSSTaxability="optimize"` is set, all max/min operations in Eqs. (5.48)–(5.50) are encoded exactly within the LP as a mixed-integer program (MIP) using the continuous variables $p_n^{\sigma,\text{lo}}$, $p_n^{\sigma,\text{hi}}$, $p_n^{\sigma,\text{min}}$, and t_n^σ , together with two binary variables z_{0n}^σ and z_{1n}^σ per year n for which $\sum_i \bar{\zeta}_{in} > 0$; in years with no Social Security income all four continuous variables and both binaries are trivially fixed to zero and excluded from the MIP enumeration. The non-negativity of $p_n^{\sigma,\text{lo}}$ and $p_n^{\sigma,\text{hi}}$ combined with the lower-bound constraints $p_n^{\sigma,\text{lo}} \geq \Pi_n - \mathcal{P}_n^{\text{lo}}$ and $p_n^{\sigma,\text{hi}} \geq \Pi_n - \mathcal{P}_n^{\text{hi}}$ enforce the $\max(0, \cdot)$ operations; because maximizing spending minimizes taxes, the optimizer pushes these variables to their lower bounds at the optimum. The two $\min(\cdot, \cdot)$ operations in Eq. (5.50) are formulated via big- $\mathcal{M}$ lower-bound constraints. For $p_n^{\sigma,\text{min}} = \min(\Delta\mathcal{P}, p_n^{\sigma,\text{lo}})$, the upper bounds

$$p_n^{\sigma,\text{min}} \leq \Delta\mathcal{P}, \quad p_n^{\sigma,\text{min}} \leq p_n^{\sigma,\text{lo}} \quad (5.51)$$

are complemented by binary-controlled lower bounds that force the optimizer to the tighter constraint:

$$p_n^{\sigma,\text{min}} - \mathcal{M} z_{0n}^\sigma \geq \Delta\mathcal{P} - \mathcal{M}, \quad p_n^{\sigma,\text{min}} - p_n^{\sigma,\text{lo}} + \mathcal{M} z_{0n}^\sigma \geq 0. \quad (5.52)$$

Analogously, for $t_n^\sigma = \min(0.85 \sum_i \bar{\zeta}_{in}, 0.5 p_n^{\sigma,\text{min}} + 0.85 p_n^{\sigma,\text{hi}})$:

$$t_n^\sigma \leq 0.85 \sum_i \bar{\zeta}_{in}, \quad t_n^\sigma \leq 0.5 p_n^{\sigma,\text{min}} + 0.85 p_n^{\sigma,\text{hi}}, \quad (5.53)$$

$$t_n^\sigma - \mathcal{M} z_{1n}^\sigma \geq 0.85 \sum_i \bar{\zeta}_{in} - \mathcal{M}, \quad t_n^\sigma - 0.5 p_n^{\sigma,\text{min}} - 0.85 p_n^{\sigma,\text{hi}} + \mathcal{M} z_{1n}^\sigma \geq 0. \quad (5.54)$$

In this formulation t_n^σ is an LP variable rather than the fixed parameter $R_n \sum_i \bar{\zeta}_{in}$ used in SC-loop mode. Moving t_n^σ to the left-hand side, Eq. (5.42) becomes

$$\begin{aligned}
e_n + \sum_{t=0}^{N_t-1} f_{tn} - t_n^\sigma - \sum_i [w_{i1n} + x_{in}] \\
- \sum_{i,k \neq 0} [(b_{i0n} - w_{i0n} + d_{in}) \alpha_{i0kn} \tau_{kn}] = \mathcal{A}_n^x + \sum_i [\omega_{in} + v_{in} + \nu_{in} + \bar{\pi}_{in}] \\
+ 0.5 \sum_{i,k \neq 0} \alpha_{i0kn} \tau_{kn} K_{i0n}, \tag{5.55}
\end{aligned}$$

and $R_n = t_n^\sigma / \sum_i \bar{\zeta}_{in}$ is recovered from the LP solution for reporting. There is no circular dependency: provisional income Π_n is constructed from non-SS ordinary income, Roth conversions, Q_n dividends, and the fixed half of SS ($0.5 \sum_i \bar{\zeta}_{in}$), so t_n^σ does not appear in Π_n .

5.1.19 Medicare brackets and costs

Annual Medicare costs m_n include the income-related monthly adjusted amount commonly known as IRMAA. As this additional adjustment is a piecewise constant function, it can be computed using binary variables and mixed-integer linear programming. In the current tax code, this adjustment depends on the modified adjusted gross income (MAGI) $\mathbb{G}$ from 2 years earlier. For the MAGI, we use the AGI basis $\mathbb{G}_{(n-2)} = G_{(n-2)} + Q_{(n-2)} + e_{(n-2)}$, i.e., gross taxable ordinary income plus dividends and capital gains plus the standard exemption. Only the *taxable* portion of Social Security enters IRMAA MAGI (it is already contained in $G_{(n-2)} + e_{(n-2)}$); the non-taxable portion is not added back. If the plan has its oldest individual currently either 64 or 65 years old, MAGI values for previous years must be provided by the user to complete the calculations.

Including regular premiums, there are $N_q = 6$ brackets of MAGI indexed for inflation defined with annual cumulative Medicare costs of $\bar{\mathcal{C}}_{qn} = \gamma_n \mathcal{C}_q$ also adjusted for inflation. These segments are separated by $N_q - 1$ thresholds $\bar{\mathcal{L}}_{qn} = \gamma_n \mathcal{L}_q$. Points $(\bar{\mathcal{L}}_{qn}, \bar{\mathcal{C}}_{qn})$ form a piecewise constant function that could be modeled using a special ordered set of type 1 (SOS1). Points $(\mathcal{L}_q, \mathcal{C}_q)$ are shown in Fig. 5.1. When $\mathbb{G}_{(n-2)} \leq \bar{\mathcal{L}}_{0n}$, the value $\bar{\mathcal{C}}_{0n}$ represents the (inflation-adjusted) base premium for Medicare Part B in year n , while higher q values include the IRMAA additional costs.

To model the IRMAA brackets we introduce binary variables $z_{qn}^m \in \{0, 1\}$ (one per bracket) and continuous variables $h_{qn} \geq 0$ for the MAGI assigned to the selected

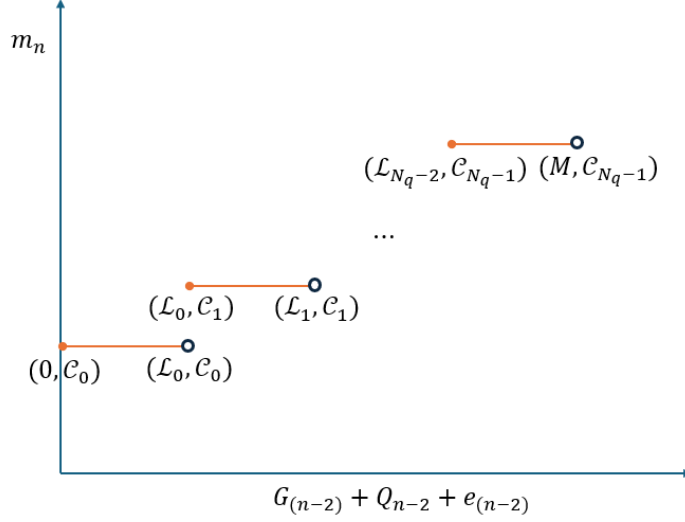


Figure 5.1: Modeling a piecewise constant monotonically increasing function. The x -axis represents the MAGI from two years ago, while m_n is the Medicare adjusted premiums. A binary variable z_q^m is associated with each segment q .

bracket q . Big- $\mathcal{M}$ is used only for the upper bound of the top bracket (see below). We impose an SOS1 selection on binary z_{qn}^m as

$$\sum_q z_{qn}^m = 1, \quad (5.56)$$

and the disaggregation constraints

$$\sum_{q=0}^{N_q-1} h_{qn} = \mathbb{G}_{(n-2)}, \quad (5.57)$$

$$\begin{aligned} 0 \leq h_{0n} &\leq \bar{\mathcal{L}}_{0n} z_{0n}^m, \\ \bar{\mathcal{L}}_{(q-1)n} z_{qn}^m \leq h_{qn} &\leq \bar{\mathcal{L}}_{qn} z_{qn}^m, \quad q \in \{1, \dots, N_q - 2\}, \\ \bar{\mathcal{L}}_{(N_q-2)n} z_{(N_q-1)n}^m \leq h_{(N_q-1)n} &\leq \mathcal{M} \gamma_n z_{(N_q-1)n}^m. \end{aligned} \quad (5.58)$$

for all $n \in \{n_m, \dots, N_n - 1\}$, where n_m is the index year when Medicare starts for any individual i in the plan. The upper bound on the last bracket is necessary: when $z_{(N_q-1)n}^m = 0$ (the top bracket is not selected), the lower bound alone only gives $h_{(N_q-1)n} \geq 0$, so the solver could assign positive MAGI to an inactive bracket. The constraint $h_{(N_q-1)n} \leq \mathcal{M} \gamma_n z_{(N_q-1)n}^m$ forces $h_{(N_q-1)n} \leq 0$ when $z_{(N_q-1)n}^m = 0$; with

$h_{qn} \geq 0$ we obtain $h_{(N_q-1)n} = 0$ for the inactive bracket. When $z_{(N_q-1)n}^m = 1$, $\mathcal{M}\gamma_n$ is chosen large enough to not bind (e.g. of order 10^8 in real units). As this applies to each individual eligible for Medicare, values $\bar{\mathcal{L}}_{qn}$ and $\bar{\mathcal{C}}_{qn}$ can include the accounting details for when both spouses are eligible and/or alive. For each year $n \geq n_m$, we can then express the Medicare costs m_n as

$$m_n = \sum_{q=0}^{N_q-1} z_{qn}^m \bar{\mathcal{C}}_{qn}, \quad (5.59)$$

as $z_{qn}^m = \delta_{q,q'}$ can only select a unique bracket q' . For $n \geq 2$, Eq. (5.57) can be expanded using Eqs. (4.24) and (4.25) as

$$\begin{aligned} \sum_{q=0}^{N_q-1} h_{qn} &= \mathcal{A}_{(n-2)}^x + \mathcal{A}_{(n-2)}^c + \sum_i [\omega_{i(n-2)} + v_{i(n-2)} + \nu_{i(n-2)} \\ &\quad + R_{(n-2)} \bar{\zeta}_{i(n-2)} + \bar{\pi}_{i(n-2)}] \\ &\quad - e_{(n-2)} - \sum_i [w_{i1(n-2)} + x_{i(n-2)}] \\ &\quad - \sum_i [(b_{i0(n-2)} - w_{i0(n-2)} + d_{i(n-2)}) \\ &\quad \cdot \left(\sum_{k \neq 0} [\alpha_{i0k(n-2)} \tau_{k(n-2)}] + \mu \alpha_{i00(n-2)} \right)] \\ &\quad + \sum_i [w_{i0(n-2)} \alpha_{i00(n-2)} \\ &\quad \cdot \max(0, \tau_{0 \max(0, n-3)} - \mu)] \\ &\quad + 0.5 \sum_i \left[\kappa_{i0(n-2)} \left(\sum_{k \neq 0} [\alpha_{i0k(n-2)} \tau_{k(n-2)}] + \mu \alpha_{i00(n-2)} \right) \right]. \end{aligned}$$

Collecting variables on the left-hand side yields

$$\begin{aligned}
& \sum_{q=0}^{N_q-1} h_{qn} + e_{(n-2)} + \sum_i [w_{i1(n-2)} + x_{i(n-2)}] \\
& + \sum_i \left[(b_{i0(n-2)} - w_{i0(n-2)} + d_{i(n-2)}) \right. \\
& \quad \cdot \left. \left(\sum_{k \neq 0} [\alpha_{i0k(n-2)} \tau_{k(n-2)}] + \mu \alpha_{i00(n-2)} \right) \right] \\
& - \sum_i [w_{i0(n-2)} \alpha_{i00(n-2)} \max(0, \tau_{0 \max(0, n-3)} - \mu)] \\
& = \mathcal{A}_{(n-2)}^x + \mathcal{A}_{(n-2)}^c + \sum_i [\omega_{i(n-2)} + v_{i(n-2)} + \nu_{i(n-2)} + R_{(n-2)} \bar{\zeta}_{i(n-2)} + \bar{\pi}_{i(n-2)}] \\
& + 0.5 \sum_i \left[\kappa_{i0(n-2)} \left(\sum_{k \neq 0} [\alpha_{i0k(n-2)} \tau_{k(n-2)}] + \mu \alpha_{i00(n-2)} \right) \right].
\end{aligned} \tag{5.60}$$

Eq. (5.57) is expanded using Eqs. (4.24) and (4.25) to yield a linear equality in the decision variables; it is implemented directly in the constraint matrix. For indices at the beginning of the time sequence, we use $\tau_{0, \max(0, n-x)}$, when $x > n$. IRMAA MAGI is on the AGI basis, so only the *taxable* portion $R_{(n-2)} \bar{\zeta}_{i(n-2)}$ of Social Security enters; the non-taxable remainder is not added back. When Social Security taxability is itself optimized (`withSSTaxability="optimize"`), the taxable amount is the LP variable $t_{(n-2)}^\sigma$ rather than the parameter $R_{(n-2)} \bar{\zeta}_{i(n-2)}$.

For plans where the year index when Medicare first starts $n_m < 2$, we will request and use user-provided values for the first years' MAGIs as $\mathbb{G}_{(-1)}$ and possibly $\mathbb{G}_{(-2)}$ and set

$$\sum_{q=0}^{N_q-1} h_{qn} = \mathbb{G}_{(n-2)}, \quad n \in \{n_m, \dots, 1\}, \tag{5.61}$$

where $n_m \geq 0$. Using Eq. (5.58), we can also check for bounds and add constraints directly on z_{q0}^m and possibly z_{q1}^m .

5.1.20 ACA marketplace premium constraints

When `withACA="optimize"` and ACA is enabled ($\text{SLCSP} > 0$, $n_{\text{aca}} > 0$), ACA bracket selection and net premium cost are modeled within the optimization. ACA

uses *current-year* MAGI $\mathbb{G}_n$ (no two-year lag like Medicare). Let $N_r = 7$ (brackets 0–5: FPL-based; bracket 6: above 400% FPL).

SOS1 selection:

$$\sum_{r=0}^{N_r-1} z_{rn}^{\text{aca}} = 1, \quad n \in \{0, \dots, n_{\text{aca}} - 1\}. \quad (5.62)$$

MAGI decomposition:

$$\sum_{r=0}^{N_r-1} h_{rn}^{\text{aca}} = \mathbb{G}_n, \quad (5.63)$$

where $\mathbb{G}_n$ is the current-year MAGI (same structure as in Eq. (5.60) but with n instead of $n - 2$).

Bracket bounds (analogous to Eq. (5.58) with Big- $\mathcal{M}$ for the last bracket):

$$\begin{aligned} 0 \leq h_{0n}^{\text{aca}} &\leq \bar{\mathcal{L}}_{0n}^{\text{aca}} z_{0n}^{\text{aca}}, \\ \bar{\mathcal{L}}_{(r-1)n}^{\text{aca}} z_{rn}^{\text{aca}} &\leq h_{rn}^{\text{aca}} \leq \bar{\mathcal{L}}_{rn}^{\text{aca}} z_{rn}^{\text{aca}}, \quad r \in \{1, \dots, N_r - 2\}, \\ \bar{\mathcal{L}}_{(N_r-2)n}^{\text{aca}} z_{(N_r-1)n}^{\text{aca}} &\leq h_{(N_r-1)n}^{\text{aca}} \leq \mathcal{M} \gamma_n z_{(N_r-1)n}^{\text{aca}}. \end{aligned} \quad (5.64)$$

Cost constraint: For brackets 0–5, net cost equals the applicable percentage times MAGI in that bracket. For bracket 6 (above 400% FPL), no subsidy applies and net cost equals the full SLCSP:

$$m_n^{\text{aca}} = \sum_{r=0}^5 \bar{\mathcal{C}}_r^{\text{aca}} h_{rn}^{\text{aca}} + \bar{S}_n^{\text{aca}} z_{6n}^{\text{aca}}, \quad n \in \{0, \dots, n_{\text{aca}} - 1\}. \quad (5.65)$$

For $n \geq n_{\text{aca}}$, $m_n^{\text{aca}} = 0$ (Medicare-eligible; no ACA cost). The variable m_n^{aca} is added to the cash-flow constraint Eq. (5.24) alongside m_n when `withACA="optimize"` is active.

6. Mapping of decision variables

At this point, one can use one of the many algebraic modeling languages such as AMPL, GAMS, MOSEK, AIMMS, and Gurobi, and code the equations above using that language, but most of these applications are proprietary and require a license and additional software installation. These languages allow the problem to be stated at a high level and steps to cast the problem in a form suitable for solution are performed automatically. There are also object-oriented language extensions, such as Python's Pyomo, that can ease the process of solving these problems. For completeness, however, we present here a simple index mapping approach that allows solving this problem using a generic linear programming solver.

Using a simple interface for mapping sparse objects to dense ones, the approach described here has been successfully tested with both the HiGHS open-source solver and the MOSEK proprietary solver. To cast the problem in a form suitable for a linear programming solver, we will use a single block vector represented by the array $y[q()]$ with index-mapping functions $q()$. While this process can be achieved using slicing and reshaping in some programming languages, we will present a generic approach suitable for most programming languages. The detailed approach presented here also allows us to determine the size of the problem to solve. We proceed alphabetically for all variables, and continue to use the convention of having index 0 for representing the first element.

To bring all variables in a single block vector, we will simply use two generic index mapping functions defined as

$$q_*(C, \ell_1, \ell_2, \ell_3; N_1, N_2, N_3) := C + \ell_1 N_2 N_3 + \ell_2 N_3 + \ell_3, \quad (6.1)$$

and

$$q_C(C, N_1, N_2, N_3) := C + N_1 N_2 N_3, \quad (6.2)$$

with the constraint that $0 \leq \ell_i < N_i$.

Account balances (b) For storing the savings account balances appropriately, variable b_{ijn} needs to have one more entry ($N_n + 1$) to store the end-of-life estate value. Therefore, we use

$$y[q_b(i, j, n)] = b_{ijn}, \quad (6.3)$$

where

$$q_b(i, j, n) = q_*(C_b, i, j, n; N_i, N_j, N_n + 1), \quad (6.4)$$

and where n exceptionally runs from 0 to N_n inclusively, and therefore q_b runs from $C_b = 0$ to $C_d - 1$, where

$$C_d = q_C(C_b, N_i, N_j, N_n + 1) = N_i N_j (N_n + 1).$$

Surplus deposits (d) For surplus deposits in the taxable savings accounts d_{in} we will use

$$y[q_d(i, n)] = d_{in}, \quad (6.5)$$

where

$$q_d(i, n) = q_*(C_d, i, n, 0; N_i, N_n, 1), \quad (6.6)$$

with q_d running from C_d to $C_e - 1$, where

$$C_e = q_C(C_d, N_i, N_n, 1) = N_i(N_j(N_n + 1) + N_n).$$

Standard exemption (e) For the standard exemption e_n we will use

$$y[q_e(n)] = e_n, \quad (6.7)$$

where

$$q_e(n) = q_*(C_e, n, 0, 0; N_n, 1, 1) = C_e + n, \quad (6.8)$$

with q_e running from C_e to $C_f - 1$, where

$$C_f = q_C(C_e, N_n, 1, 1) = N_i(N_j(N_n + 1) + N_n) + N_n.$$

Tax bracket amounts (f) For tax bracket amounts f_{tn} we will use

$$y[q_f(t, n)] = f_{tn}, \quad (6.9)$$

where

$$q_f(t, n) = q_*(C_f, t, n, 0; N_t, N_n, 1) \quad (6.10)$$

with q_f running from C_f to $C_g - 1$, where

$$C_g = q_C(C_f, N_t, N_n, 1) = N_i(N_j(N_n + 1) + N_n) + (N_t + 1)N_n.$$

Net spending (g) For net spending g_n we will use

$$y[q_g(n)] = g_n, \quad (6.11)$$

where

$$q_g(n) = q_*(C_g, n, 0, 0; N_n, 1, 1) = C_g + n, \quad (6.12)$$

with q_g running from C_g to $C_h - 1$, where

$$C_h = q_C(C_g, N_n, 1, 1) = N_i(N_j(N_n + 1) + N_n) + (N_t + 2)N_n.$$

MAGI bracket portions (h) For h_{qn} we will use

$$y[q_h(q, n)] = h_{qn}, \quad (6.13)$$

where

$$q_h(q, n) = q_*(C_h, q, n, 0; N_q, N_n - n_m, 1), \quad (6.14)$$

with q_h running from C_h to $C_m - 1$, where

$$C_m = q_C(C_h, N_q, N_n - n_m, 1) = N_i(N_j(N_n + 1) + N_n) + (N_t + 2)N_n + N_q(N_n - n_m).$$

Medicare costs (m) For Medicare costs m_n we will use

$$y[q_m(n)] = m_n, \quad (6.15)$$

where

$$q_m(n) = q_*(C_m, n, 0, 0; N_n, 1, 1) = C_m + n, \quad (6.16)$$

with q_m running from C_m to $C_{s_0} - 1$, where

$$C_{s_0} = q_C(C_m, N_n, 1, 1) = N_i(N_j(N_n + 1) + N_n) + (N_t + 3)N_n + N_q(N_n - n_m).$$

LTCG bracket allocations (q) For LTCG bracket allocation variables q_{pn} we will use

$$y[q_q(p, n)] = q_{pn}, \quad (6.17)$$

where

$$q_q(p, n) = q_*(C_q, p, n, 0; N_p, N_n, 1), \quad (6.18)$$

with q_q running from C_q to $C_s - 1$, where $C_q = C_{s_0}$ (the cumulative index after Medicare) and

$$C_s = q_C(C_q, N_p, N_n, 1) = N_i(N_j(N_n + 1) + N_n) + (N_t + 3 + N_p)N_n + N_q(N_n - n_m).$$

Surplus (s) Surplus can be generated if big-ticket items are received (inheritance, sale of a house, etc.) or due to RMDs. Surplus s is then deposited to taxable savings accounts according to variable η . We will use

$$y[q_s(n)] = s_n, \quad (6.19)$$

where

$$q_s(n) = q_*(C_s, n, 0, 0; N_n, 1, 1) = C_s + n, \quad (6.20)$$

with q_s running from C_s to $C_w - 1$, where

$$C_w = q_C(C_s, N_n, 1, 1) = N_i(N_j(N_n + 1) + N_n) + (N_t + 4 + N_p)N_n + N_q(N_n - n_m).$$

Withdrawals (w) For withdrawals w_{ijn} we will use

$$y[q_w(i, j, n)] = w_{ijn}, \quad (6.21)$$

where

$$q_w(i, j, n) = q_*(C_w, i, j, n; N_i, N_j, N_n), \quad (6.22)$$

with q_w running from C_w to $C_x - 1$, where

$$C_x = q_C(C_w, N_i, N_j, N_n) = N_i(N_j(2N_n + 1) + (N_t + 4 + N_p)N_n) + N_q(N_n - n_m).$$

Roth conversions ($\mathbf{x}$) Finally, for Roth conversions x_{in} we will use

$$y[q_x(i, n)] = x_{in}, \quad (6.23)$$

where

$$q_x(i, n) = q_*(C_x, i, n, 0; N_i, N_n, 1), \quad (6.24)$$

with q_x running from C_x to $C_* - 1$, where

$$C_* = q_C(C_x, N_i, N_n, 1) = N_i(N_j(2N_n + 1) + (N_t + 5 + N_p)N_n) + N_q(N_n - n_m). \quad (6.25)$$

With $N_i = 2$, $N_j = 4$, $N_k = 4$, $N_t = 7$, $N_p = 3$, Eq. (6.25) yields $C_* = 46N_n + 8 + N_q(N_n - n_m)$ entries in y (for couples, $(N_t + 5 + N_p) = (N_t + N_i + 3 + N_p)$). If $n_m = N_n$, then $N_n - n_m = 0$ and the IRMAA SOS block for h_{qn} is empty, so $C_* = 46N_n + 8$. The $N_p N_n$ LTCG bracket allocations q_{pn} are already included in this count. If the time resolution is increased to months, the number of variables scales accordingly. Adding binary variables z_{nz}^x ($z \in \{0, 1, 2, 3\}$) always contributes $4N_n$ binary variables. Optional MIP families add further binaries depending on which "optimize" flags are active: $N_q(N_n - n_m)$ for z_{qn}^m (Medicare IRMAA), $N_{aca}(N_n - n_{aca})$ for z_{rn}^a (ACA brackets), $2N_n^{ss}$ for $z_{0n}^\sigma, z_{1n}^\sigma$ (SS taxability), $2N_n$ for z_{0n}^l, z_{1n}^l (LTCG regime), and N_n for z_n^j (NIIT threshold), where N_n^{ss} is the number of years with positive SS income. When all five optional families are active simultaneously, the total binary count can reach approximately 400 for a 30-year couple plan.

6.1 Reverse mapping of indices

The inverse functions for the index-mapping functions will be derived for the most complex case encountered in this paper. If we have

$$z = q_*(C, i, j, k, n; N_i, N_j, N_k, N_n) := C + iN_j N_k N_n + jN_k N_n + kN_n + n, \quad (6.26)$$

then $(i, j, k, n) = q_*^{-1}(z; N_i, N_j, N_k, N_n, C)$ is obtained from

$$\begin{aligned} n &= \text{mod}(\text{mod}(\text{mod}(z - C, N_j N_k N_n), N_k N_n), N_n), \\ k &= \text{mod}(\text{mod}(z - C - n, N_j N_k N_n), N_k N_n) / N_n, \\ j &= \text{mod}(z - C - n - kN_n, N_j N_k N_n) / (N_k N_n), \\ i &= (z - C - n - kN_n - jN_k N_n) / (N_j N_k N_n). \end{aligned} \quad (6.27)$$

While this holds for all cases presented in the previous section, this can be easily simplified for cases having fewer active indices. However, some modern languages can accomplish this mapping rather easily by providing `reshape()` functions.

7. Building constraint matrices

Let's first define generic index-mapping functions I and J as

$$\begin{aligned}
 I_l(n) &= C_l + n, \\
 I_l(i, n; N_n) &= C_l + iN_n + n, \\
 I_l(i, j, n; N_j, N_n) &= C_l + iN_jN_n + jN_n + n, \\
 \dots &= \dots
 \end{aligned} \tag{7.1}$$

and so on, which would cumulatively increase row count C_l at each new instance l , similar to how we proceeded in the previous section. This allows us to build rectangular matrices by iteratively adding rows. These constraint matrices have C_* (defined in Eq. (6.25)) columns but can have fewer rows, forming an underdetermined system to be optimized using linear programming. Function J is defined similarly for equality constraints, while I is used for building the rows of the matrix containing the inequality constraints. Additional indices found in columns and not in rows imply the existence of multiple elements on that row.

7.1 Inequality constraints

Inequality constraints can be upper or lower bounds expressed in matrix form as $\ell \leq A_u y \leq u$. When ℓ is not specified it is assumed 0, and an unspecified u implies ∞ . Most if not all inequalities will be expressed as upper bounds.

Required minimum distributions (RMDs) We rewrite the inequality constraint on required minimum distributions Eq. (5.5) using matrix $A_u y \leq u$ starting

with the following $N_i N_n$ rows,

$$\begin{aligned}
A_u[I_0(i, n), q_w(i, 1, n)] &= -1 \\
A_u[I_0(i, n), q_b(i, 1, n)] &= \rho_{in}, \\
u[I_0(i, n)] &= 0, \\
\forall i \in \{0, \dots, N_i - 1\}, \\
\forall n \in \{0, \dots, N_n - 1\},
\end{aligned} \tag{7.2}$$

and all other elements in the same rows of A_u being 0. Notice that while b has $N_n + 1$ elements, the constraints for b go from 0 to $N_n - 1$ as there is no RMD required in the last year of the plan N_n , when all individuals have passed. Years before RMDs where $\rho_{in} = 0$ just add a trivial constraint. See Eq. (6.4).

Ordinary income tax brackets Similarly, we add $N_t N_n$ more rows to matrix $A_u y \leq u$ to express the inequality constraint in Eq. (5.6) setting an upper limit on amounts f_{tn} ,

$$\begin{aligned}
A_u[I_1(t, n), q_f(t, n)] &= 1, \\
u[I_1(t, n)] &= \bar{\Delta}_{tn}, \\
\forall t \in \{0, \dots, N_t - 1\}, \\
\forall n \in \{0, \dots, N_n - 1\},
\end{aligned} \tag{7.3}$$

and all other elements in the same rows of A_u being 0.

Standard exemption For $e_n \leq \bar{\sigma}_n$ we add

$$\begin{aligned}
A_u[I_2(n), q_e(n)] &= 1, \\
u[I_2(n)] &= \bar{\sigma}_n, \\
\forall n \in \{0, \dots, N_n - 1\}.
\end{aligned} \tag{7.4}$$

Non-negativity of e_n provides the lower bound.

Withdrawal limits For $w_{ijn} + \delta_{j,1}x_{in} \leq b_{ijn}$ we add:

$$\begin{aligned}
A_u[I_3(i, j, n), q_w(i, j, n)] &= 1, \\
A_u[I_3(i, j, n), q_x(i, n)] &= \delta_{j,1}, \\
A_u[I_3(i, j, n), q_b(i, j, n)] &= -1, \\
u[I_3(i, j, n)] &= 0, \\
&\forall i \in \{0, \dots, N_i - 1\}, \\
&\forall j \in \{0, \dots, N_j - 1\}, \\
&\forall n \in \{0, \dots, N_n - 1\} \text{ } (n < n_d \text{ for } i = i_d).
\end{aligned}$$

For a couple, the upper index for the first-to-pass individual i_d is $n_d - 1$ rather than $N_n - 1$, since posthumous constraints zero out all account activities.

Posthumous account activities For $n \geq n_d$ and the first-to-pass individual i_d , we set

$$\begin{aligned}
A_u[I_4(j, n), q_w(i_d, j, n)] &= 1, \\
u[I_4(j, n)] &= 0, \\
A_u[I_5(n), q_d(i_d, n)] &= 1, \\
u[I_5(n)] &= 0, \\
A_u[I_6(n), q_x(i_d, n)] &= 1, \\
u[I_6(n)] &= 0, \\
&\forall j \in \{0, \dots, N_j - 1\}, \\
&\forall n \in \{n_d, \dots, N_n - 1\}.
\end{aligned}$$

Roth conversion limit If a maximum conversion x_{max} is specified, we add the bound $\sum_i x_{in} \leq x_{max}$ for all n with

$$\begin{aligned}
A_u[I_7(n), q_x(i, n)] &= 1, \\
u[I_7(n)] &= x_{max}, \\
&\forall i \in \{0, \dots, N_i - 1\}, \\
&\forall n \in \{0, \dots, N_n - 1\}.
\end{aligned}$$

This maximum is applied to the combined conversions of both individuals in each year.

Medicare brackets Medicare's MAGI brackets use binary variables z_{qn}^m and the disaggregated MAGI variables h_{qn} . We set upper bounds for each bracket using Eq. (5.58) as

$$\begin{aligned}
A_u[I_{15}(q, n), q_h(q, n)] &= 1, \\
A_u[I_{15}(q, n), q_{z^m}(q, n)] &= -\bar{\mathcal{L}}_{qn}, \\
u[I_{15}(q, n)] &= 0, \\
&\forall q \in \{0, \dots, N_q - 2\}, \\
&\forall n \in \{n_m, \dots, N_n - 1\},
\end{aligned} \tag{7.5}$$

the upper bound for the last bracket ($q = N_q - 1$) using big- $\mathcal{M}$ as

$$\begin{aligned}
A_u[I_{15}(N_q - 1, n), q_h(N_q - 1, n)] &= 1, \\
A_u[I_{15}(N_q - 1, n), q_{z^m}(N_q - 1, n)] &= -\mathcal{M}\gamma_n, \\
u[I_{15}(N_q - 1, n)] &= 0, \\
&\forall n \in \{n_m, \dots, N_n - 1\},
\end{aligned} \tag{7.6}$$

and the lower bounds as

$$\begin{aligned}
A_u[I_{16}(q, n), q_h(q, n)] &= -1, \\
A_u[I_{16}(q, n), q_{z^m}(q, n)] &= \bar{\mathcal{L}}_{(q-1)n}, \\
u[I_{16}(q, n)] &= 0, \\
&\forall q \in \{1, \dots, N_q - 1\}, \\
&\forall n \in \{n_m, \dots, N_n - 1\}.
\end{aligned} \tag{7.7}$$

7.2 Equality constraints

Account balances For the equality constraints on account balances expressed in Eq. (5.23), we define an equality constraint matrix $A_e y = v$ starting with $N_i N_j N_n$

rows as

$$\begin{aligned}
A_e[J_0(i, j, n), q_b(i, j, n + 1)] &= 1, \\
A_e[J_0(i, j, n), q_b(i, j, n)] &= -(1 + \mathcal{T}_{ijn}), \\
A_e[J_0(i, j, n), q_x(i, n)] &= -(\delta_{j,2} - \delta_{j,1})(1 + \mathcal{T}_{ijn}), \\
A_e[J_0(i, j, n), q_w(i, j, n)] &= (1 + \mathcal{T}_{ijn}), \\
A_e[J_0(i, j, n), q_d(i, n)] &= -\delta_{j,0}(1 + \mathcal{T}_{ijn}), \\
&\quad \forall i \in \{0, \dots, N_i - 1\}, \\
&\quad \forall j \in \{0, \dots, N_j - 1\}, \\
&\quad \forall n \in \{0, \dots, N_n - 1\},
\end{aligned} \tag{7.8}$$

where v is

$$v[J_0(i, j, n)] = \kappa_{ijn}(1 + \mathcal{T}_{ijn}/2). \tag{7.9}$$

The initial account balances expressed in Eq. (5.17) are imposed through

$$\begin{aligned}
A_e[J_1(i, j), q_b(i, j, 0)] &= 1, \\
v[J_1(i, j)] &= \beta_{ij},
\end{aligned} \tag{7.10}$$

$$\begin{aligned}
&\quad \forall i \in \{0, \dots, N_i - 1\}, \\
&\quad \forall j \in \{0, \dots, N_j - 1\},
\end{aligned} \tag{7.11}$$

leading to $N_i N_j$ additional rows to A_e .

Medicare bracket selection The SOS1 selection of Eq. (5.56) is encoded as

$$\begin{aligned}
A_e[J_2(n), q_{zm}(q, n)] &= 1, \\
v[J_2(n)] &= 1, \\
&\quad \forall q \in \{0, \dots, N_q - 1\}, \\
&\quad \forall n \in \{n_m, \dots, N_n - 1\}.
\end{aligned} \tag{7.12}$$

MAGI disaggregation Using Eqs. (4.24) and (4.25), the equality $\sum_q h_{qn} = \mathbb{G}_{(n-2)}$ is expanded and implemented as

$$\begin{aligned}
A_e[J_3(n), q_h(q, n)] &= 1, \\
A_e[J_3(n), q_e(n-2)] &= -1, \\
A_e[J_3(n), q_w(i, 1, n-2)] &= -1, \\
A_e[J_3(n), q_x(i, n-2)] &= -1, \\
A_e[J_3(n), q_b(i, 0, n-2)] &= -\mu\alpha_{i00(n-2)} \\
&\quad - \sum_{k \neq 0} \alpha_{i0k(n-2)} \tau_{k(n-2)}, \\
A_e[J_3(n), q_d(i, n-2)] &= -\mu\alpha_{i00(n-2)} \\
&\quad - \sum_{k \neq 0} \alpha_{i0k(n-2)} \tau_{k(n-2)}, \\
A_e[J_3(n), q_w(i, 0, n-2)] &= \mu\alpha_{i00(n-2)} + \sum_{k \neq 0} \alpha_{i0k(n-2)} \tau_{k(n-2)} \\
&\quad - \alpha_{i00(n-2)} \max(0, \tau_{0 \max(0, n-3)} - \mu), \\
v[J_3(n)] &= \mathcal{A}_{n-2}^x + \mathcal{A}_{n-2}^c + \sum_i [\omega_{i(n-2)} + \nu_{i(n-2)} + \nu_{i(n-2)} \\
&\quad + \bar{\zeta}_{i(n-2)} + \bar{\pi}_{i(n-2)}] \\
&\quad + 0.5 \sum_i \left[\kappa_{i0(n-2)} \left(\mu\alpha_{i00(n-2)} + \sum_{k \neq 0} \alpha_{i0k(n-2)} \tau_{k(n-2)} \right) \right], \\
&\quad \forall i \in \{0, \dots, N_i - 1\}, \\
&\quad \forall k \in \{0, \dots, N_k - 1\}, \\
&\quad \forall q \in \{0, \dots, N_q - 1\}, \\
&\quad \forall n \in \{\max(n_m, 2), \dots, N_n - 1\}. \tag{7.13}
\end{aligned}$$

When $n_m < 2$, we instead set $v[J_3(n)] = \mathbb{G}_{(-1)}$ and $\mathbb{G}_{(-2)}$, which are user-provided MAGI values for the years before the plan started.

Medicare costs Binary variables z_{qn}^m express the active IRMAA bracket. The costs for Medicare are obtained using Eq. (5.59):

$$\begin{aligned}
A_e[J_4(n), q_m(n)] &= 1, \\
A_e[J_4(n), q_{zm}(q, n)] &= -\bar{C}_{qn}, \\
v[J_4(n)] &= 0, \\
\forall q \in \{0, \dots, N_q - 1\}, \\
\forall n \in \{n_m, \dots, N_n - 1\}.
\end{aligned}$$

Net spending For the equality constraint on net spending expressed in Eq. (5.24), we add N_n more rows to $A_e y = v$ as

$$\begin{aligned}
A_e[J_5(n), q_g(n)] &= 1, \\
A_e[J_5(n), q_m(n)] &= 1, \\
A_e[J_5(n), q_s(n)] &= 1, \\
A_e[J_5(n), q_f(t, n)] &= \theta_{tn}^x, \\
A_e[J_5(n), q_w(i, j, n)] &= -1 + 0.1 \delta_{j,1} (1 - \mathcal{H}(n - n_{i,59\frac{1}{2}})), \\
A_e[J_5(n), q_q(1, n)] &= 0.15, \\
A_e[J_5(n), q_q(2, n)] &= 0.20, \\
\forall t \in \{0, \dots, N_t - 1\}, \\
\forall i \in \{0, \dots, N_i - 1\}, \\
\forall j \in \{0, \dots, N_j - 1\}, \\
\forall n \in \{0, \dots, N_n - 1\},
\end{aligned}$$

where $q_q(p, n)$ is the flat index of q_{pn} in the solution vector, and v is

$$v[J_5(n)] = \sum_{*=x,c,f} \mathcal{A}_n^* + \sum_i \Lambda_{in}^\pm + \sum_i [\omega_{in} + v_{in} + \nu_{in} + \bar{\zeta}_{in} + \bar{\pi}_{in}] - \mathcal{D}_n. \quad (7.14)$$

The condition of having a predictable net spending expressed as an equality in Eq. (5.26) adds $N_n - 1$ more rows to $A_e y = v$ as

$$\begin{aligned}
A_e[J_6(n), q_g(0)] &= -\bar{\xi}_n, \\
A_e[J_6(n), q_g(n)] &= \xi_0, \\
v[J_6(n)] &= 0, \\
\forall n \in \{1, \dots, N_n - 1\}.
\end{aligned} \quad (7.15)$$

Taxable ordinary income For the equality constraint in Eq. (5.42) establishing taxable ordinary income, we add N_n rows to $A_e y = v$ as follows. The taxable Social Security term $R_n \sum_i \bar{\zeta}_{in}$ appears on the right-hand side as a parameter updated each SC-loop iteration.

$$\begin{aligned}
A_e[J_7(n), q_e(n)] &= 1, \\
A_e[J_7(n), q_f(t, n)] &= 1, \\
A_e[J_7(n), q_w(i, 1, n)] &= -1, \\
A_e[J_7(n), q_x(i, n)] &= -1, \\
A_e[J_7(n), q_b(i, 0, n)] &= -\sum_{k \neq 0} \alpha_{i0kn} \tau_{kn}, \\
A_e[J_7(n), q_w(i, 0, n)] &= \sum_{k \neq 0} \alpha_{i0kn} \tau_{kn}, \\
A_e[J_7(n), q_d(i, n)] &= -\sum_{k \neq 0} \alpha_{i0kn} \tau_{kn}, \\
&\quad \forall t \in \{0, \dots, N_t - 1\}, \\
&\quad \forall i \in \{0, \dots, N_i - 1\}, \\
&\quad \forall k \in \{0, \dots, N_k - 1\}, \\
&\quad \forall n \in \{0, \dots, N_n - 1\},
\end{aligned} \tag{7.16}$$

with

$$v[J_7(n)] = \mathcal{A}_n^x + R_n \sum_i \bar{\zeta}_{in} + \sum_i [\omega_{in} + v_{in} + \nu_{in} + \bar{\pi}_{in}] + 0.5 \sum_i \kappa_{i0n} \sum_{k \neq 0} \alpha_{i0kn} \tau_{kn} \tag{7.17}$$

7.3 Other considerations

Beneficiaries Tax-free and tax-deferred accounts have special tax rules that allow giving part or the entire value of tax-free accounts to a spouse who can then consider it as his/her own. These accounts typically use percentages to designate beneficiaries. Let ϕ_j be the fraction of the account j that a spouse i_d wishes to leave to his/her surviving spouse i_s in year n_d , the year following the passing. To account for that

event in year n_d , Eq. (5.23) needs to be rewritten as

$$\begin{aligned}
b_{ij(n+1)} &= [1 - \delta_{n,n_d-1}\delta_{i,i_d}] \\
&\times \left\{ [b_{ijn} + \delta_{j,0}d_{in} - w_{ijn} + (\delta_{j,2} - \delta_{j,1})x_{in}] (1 + \mathcal{T}_{ijn}) + \kappa_{ijn}(1 + \mathcal{T}_{ijn}/2) \right\} \\
&+ [\phi_j \delta_{n,n_d-1} \delta_{i,i_s}] \\
&\times \left\{ [b_{idjn} + \delta_{j,0}d_{idn} - w_{idjn} + (\delta_{j,2} - \delta_{j,1})x_{idn}] (1 + \mathcal{T}_{idjn}) \right. \\
&\left. + \kappa_{idjn}(1 + \mathcal{T}_{idjn}/2) \right\}. \tag{7.18}
\end{aligned}$$

The first multiplier $[\dots]$ on the right-hand side will always be one except for i_d in year $n_d - 1$ when it will be zero. This will result in emptying all accounts for i_d for years n_d and beyond. The second special multiplier $[\dots]$ before the second set of curly braces $\{\}$ will always be zero except for the surviving spouse i_s in year $n_d - 1$, who will then inherit a fraction ϕ_j of account j that was scheduled to go into i_d 's j account at the beginning of year n_d .

Rewriting the last equation as a constraint results in

$$\begin{aligned}
&b_{ij(n+1)} \\
&- [1 - \delta_{n,n_d-1}\delta_{i,i_d}] \\
&\times \left\{ [b_{ijn} + \delta_{j,0}d_{in} - w_{ijn} + (\delta_{j,2} - \delta_{j,1})x_{in}] (1 + \mathcal{T}_{ijn}) \right\} \\
&- [\phi_j \delta_{n,n_d-1} \delta_{i,i_s}] \\
&\times \left\{ [b_{idjn} + \delta_{j,0}d_{idn} - w_{idjn} + (\delta_{j,2} - \delta_{j,1})x_{idn}] (1 + \mathcal{T}_{idjn}) \right\} \\
= & [1 - \delta_{n,n_d-1}\delta_{i,i_d}] \kappa_{ijn}(1 + \mathcal{T}_{ijn}/2) \\
&+ [\phi_j \delta_{n,n_d-1} \delta_{i,i_s}] \kappa_{idjn}(1 + \mathcal{T}_{idjn}/2). \tag{7.19}
\end{aligned}$$

We are now ready to replace Eq. (7.8) for $A_e y = v$ by

$$\begin{aligned}
A_e[J_0(i, j, n), q_b(i, j, n+1)] &= 1, \\
A_e[J_0(i, j, n), q_b(i, j, n)] &= -[1 - \delta_{n, n_d-1} \delta_{i, i_d}](1 + \mathcal{T}_{ijn}), \\
A_e[J_0(i, j, n), q_d(i, j, n)] &= -[1 - \delta_{n, n_d-1} \delta_{i, i_d}] \delta_{j,0} (1 + \mathcal{T}_{ijn}), \\
A_e[J_0(i, j, n), q_w(i, j, n)] &= [1 - \delta_{n, n_d-1} \delta_{i, i_d}](1 + \mathcal{T}_{ijn}), \\
A_e[J_0(i, j, n), q_x(i, n)] &= -[1 - \delta_{n, n_d-1} \delta_{i, i_d}] (\delta_{j,2} - \delta_{j,1}) (1 + \mathcal{T}_{ijn}), \\
\text{when } N_i = 2 \text{ and } i = i_s, \\
A_e[J_0(i, j, n), q_b(i_d, j, n)] &= -[\phi_j \delta_{n, n_d-1} \delta_{i, i_s}](1 + \mathcal{T}_{idjn}), \\
A_e[J_0(i, j, n), q_d(i_d, n)] &= -[\phi_j \delta_{n, n_d-1} \delta_{i, i_s}] \delta_{j,0} (1 + \mathcal{T}_{idjn}), \\
A_e[J_0(i, j, n), q_w(i_d, j, n)] &= [\phi_j \delta_{n, n_d-1} \delta_{i, i_s}](1 + \mathcal{T}_{idjn}), \\
A_e[J_0(i, j, n), q_x(i_d, n)] &= -[\phi_j \delta_{n, n_d-1} \delta_{i, i_s}] (\delta_{j,2} - \delta_{j,1}) (1 + \mathcal{T}_{idjn}),
\end{aligned}$$

$$\forall i \in \{0, \dots, N_i - 1\},$$

$$\forall j \in \{0, \dots, N_j - 1\},$$

$$\forall n \in \{0, \dots, N_n - 1\},$$

where v is

$$\begin{aligned}
v[J_0(i, j, n)] &= [1 - \delta_{n, n_d-1} \delta_{i, i_d}] \kappa_{ijn} (1 + \mathcal{T}_{ijn}/2) \\
&\quad + [\phi_j \delta_{n, n_d-1} \delta_{i, i_s}] \kappa_{idjn} (1 + \mathcal{T}_{idjn}/2).
\end{aligned} \tag{7.20}$$

While the last two equations may look cumbersome, their net effect is only to include a few more terms when $n = n_d - 1$.

Asset allocation ratios When asset allocation ratios α are imposed, they should also be applied to how contribution amounts κ_{ijn} are invested, such that

$$\kappa_{ijkn} = \alpha_{ijkn} \kappa_{ijn}. \tag{7.21}$$

For other allocation schemes, just substitute $\alpha_{ijkn} = \alpha_{ikn}$ or α_{kn} depending on the scheme selected.

Asset allocation has been handled easily by assuming that the accounts are always rebalanced and only using a single multiplier $\mathcal{T}$, defined as

$$\mathcal{T}_{ijn} = \sum_k \alpha_{ijkn} \tau_{kn}, \tag{7.22}$$

to compute the return on the total balance of each savings account.

Spousal deposits and withdrawals While keeping the problem linear, a simple constraint can be imposed on surplus deposits in taxable savings accounts. One can specify a spousal ratio η such as

$$d_{0n} = \eta d_{1n}. \quad (7.23)$$

A similar spousal ratio can be imposed on withdrawals from tax-deferred accounts

$$w_{01n} = \eta w_{11n}, \quad (7.24)$$

but this can cause drawing from an empty account while the other spousal account is not. Only the deposit scheme has currently been implemented in Owl.

7.4 Constraint tags and the explanation engine

Every row added to the constraint matrices carries a machine-readable *tag*: a tuple whose first element names the constraint *family* and whose remaining elements are the row’s indices, e.g. (`"rmd", i, n`) for the RMD floor of individual i in year n . Tags are inert during optimization; they exist so that the solution can be *explained*. When a plan is solved with `withDuals=True`, Owl re-solves the final linear program with all binary variables fixed at their solved values and extracts the row duals (shadow prices), row activities, and column reduced costs. The tags map each dual back to the constraint it prices, letting the explanation engine (`owlplanner.assistant.explain`, surfaced through the `explain_results` tool) answer questions such as “what does the bequest requirement cost in lifetime spending?” or “which years do RMDs force withdrawals?” directly from the mathematics of the optimization.

Each family is classified in one of three classes:

Policy User goals and legal rules (Table 7.1). A binding row of this class is reported to the user together with the marginal value of relaxing it, expressed in today’s dollars of the reported objective.

Structural Accounting identities and definitions (Table 7.2). These rows are binding by construction; their duals are valuation information — for instance, the duals of the cash-flow balances form the plan’s endogenous discount curve — and are used selectively by the explanation engine.

Artifact Big- $\mathcal{M}$ linkings, at-most-one selectors, convexification rows, and decomposition machinery (Table 7.3). These encode formulation technique rather than financial meaning and are excluded from user-facing output. Threshold

(cliff) effects that the underlying binaries model — IRMAA tiers, ACA brackets, Social Security taxability — are instead reported as primal *headroom* to the next threshold, since marginal prices cannot express discrete jumps.

The authoritative family-to-class table is `CONSTRAINT_FAMILIES` in `owlplanner/assistant/explain.py`; it must be kept in synchrony with the tags emitted in `plan.py` and with the tables below. Year indices of the Medicare and ACA families are offsets from the first Medicare- or ACA-eligible year, respectively.

Constraint rows are only half of the sensitivity picture: policy also enters the model as *bounds on decision variables*, priced by their reduced costs rather than by row duals. A companion registry (`COLUMN_FAMILIES`, same module) prices variables sitting at a policy-meaningful bound: the `maxRothConversion` cap on x_{in} , conversions pinned to zero by rule (`noRothConversions`, `startRothConversions`, the last-two-years rule), the `minTaxableBalance` safety-net floor on b_{i0n} , and the loop-mode LTCG bracket room bounds on q_{0n} and q_{1n} of Eqs. (5.29)–(5.30). At a binding upper bound, the value of one more dollar of allowance is the reduced cost converted to reported-objective units; at a binding lower bound, the same conversion prices each dollar of relaxation of the floor. Structural bounds — bracket widths $\bar{\Delta}_{tn}$, standard deductions, big- $\mathcal{M}$ caps — are deliberately not registered; bracket fill is reported through the primal bracket analysis instead. The resulting explanation payload is validated against a versioned Pydantic schema (`PlanExplanation` in `owlplanner/assistant/explain_schema.py`), which constitutes the machine-readable contract consumed by AI assistants.

Tag family	Indices	Formulation
<code>rmd</code>	(i, n)	Eq. (5.5)
<code>bequest_floor</code>	—	Eq. (8.1)
<code>profile_lo, profile_hi</code>	(n)	Eqs. (5.25)–(5.27)
<code>roth_maturation</code>	(i, n)	Sec. (5.1.10)
<code>hsa_medical_cap</code>	(n)	Sec. (2.8)
<code>state_ret_exempt_cap</code>	(n)	Eq. (5.45)
<code>ltcg_room20</code>	(n)	Eq. (5.30)

Table 7.1: Policy-class constraint tags: user goals and legal rules.

Tag family	Indices	Formulation
cash_flow	(n)	Eq. (5.24)
taxable_income	(n)	Eq. (5.42)
state_taxable_income	(n)	Eq. (5.44)
account_carryover	(i, j, n)	Eqs. (5.21)–(5.23)
withdrawal_limit	(i, j, n)	Sec. (5.1.4)
surplus_deposit	(i, n)	Eq. (5.19)
ss_age_benefit	(i, n)	Chapter 11
ltcg_gn_def	(n)	Sec. (5.1.14)
ltcg_partition_lo	(n)	Eq. (5.28)
niit_magi_def	(n)	Eq. (5.40)
irmaa_magi_def	(n)	Eqs. (5.60), (5.61)
irmaa_cost_def	(n)	Eq. (5.59)
aca_magi_def	(n)	Sec. (5.1.20)
aca_cost_def	(n)	Sec. (5.1.20)

Table 7.2: Structural-class constraint tags: accounting identities and definitions.

Tag family	Indices	Formulation
wdorder_txdef_gate	(i, n)	Eq. (5.9)
wdorder_roth_gate	(i, n)	Eq. (5.9)
wdorder_taxable_exhausted	(n)	Eq. (5.10)
wdorder_txdef_exhausted	(n)	Eq. (5.10)
wdorder_gate_monotone	(n)	Eq. (5.11)
ss_tax_plo, ss_tax_phi	(n)	Eq. (5.48)
ss_tax_pmin_ub, ss_tax_pmin_lb_cap	(n)	Eq. (5.52)
ss_tax_pmin_lb_plo	(n)	Eq. (5.52)
ss_tax_tss_ub, ss_tax_tss_lb_cap	(n)	Eq. (5.54)
ss_tax_tss_lb_formula	(n)	Eq. (5.54)
ss_age_amo	(i)	Chapter 11
ltcg_zl15_link, ltcg_zl20_link	(n)	Eqs. (5.31)–(5.32)
ltcg_room15_mip, ltcg_q0_zero	(n)	Eq. (5.34)
ltcg_room20_mip, ltcg_q01_zero	(n)	Eq. (5.35)
ltcg_zl_monotone	(n)	Sec. (5.1.14)
ltcg_partition_hi	(n)	Eq. (5.28)
niit_floor	(n)	Eq. (5.38)
niit_j_zero	(n)	Eq. (5.36)
niit_magi_cap	(n)	Eq. (5.37)
niit_surplus_cap	(n)	Eq. (5.39)
irmaa_amo	(n)	Eq. (5.56)
irmaa_bracket_lb, irmaa_bracket_ub	(n, q)	Eq. (5.58)
aca_amo	(n)	Sec. (5.1.20)
aca_bracket_lb, aca_bracket_ub	(n, r)	Sec. (5.1.20)
benders_cut	—	Sec. (3.9)

Table 7.3: Artifact-class constraint tags: formulation machinery, excluded from user-facing explanations. The withdrawal-ordering gates implement the `withdrawalOrder="taxable_first"` baseline policy of Sec. (5.1.5).

8. Objective Functions and Their Formulation

The objective function is a simple scalar defined as $c \cdot y$ that will be minimized.

8.1 Maximizing net spending

There are a few ways by which a retirement plan can be optimized. For maximizing the net spending under the constraint of a desired bequest, we introduce the following relation

$$E_n = \sum_{i,j} (1 - \nu\delta_{j,1}) b_{ijn}, \quad (8.1)$$

which is the value of the estate in nominal dollars at year n , taking into consideration the heir's marginal income tax rate ν on the ($j = 1$) tax-deferred account.

For a desired bequest ϵ_{N_n} , expressed in today's dollars, the final amount in year N_n will need to be

$$E_{N_n} = \bar{\epsilon}_{N_n} = \epsilon_{N_n} \gamma_{N_n}. \quad (8.2)$$

Fixing a bequest value amounts to adding the following constraint

$$\sum_{i,j} b_{ijn} (1 - \nu\delta_{j,1}) = \epsilon_{N_n} \gamma_{N_n}, \quad (8.3)$$

which would add one more row to $A_e y = v$ as

$$\begin{aligned} A_e[I(0), q_b(i, j, N_n)] &= (1 - \nu\delta_{j,1}) \\ v[I(0)] &= \epsilon_{N_n} \gamma_{N_n} \end{aligned} \quad (8.4)$$

$$\begin{aligned} \forall i &\in \{0, \dots, N_i - 1\}, \\ \forall j &\in \{0, \dots, N_j - 1\}, \end{aligned} \quad (8.5)$$

where $I(0)$ is used only to provide the proper row offset C_l . See Eq. (7.1).

For maximizing the net spending under the constraint of a fixed bequest, one may either maximize the first-year net spending basis g_0 (with the profile fixed by Eq. (5.26)), or maximize the sum of net spending over all years in today's dollars. In the first case, minimize the inner product $c \cdot y$ with

$$c[q_g(0)] = -1, \quad (8.6)$$

and 0 otherwise; the whole time series g_n then follows from Eq. (5.26). In the second case, when the slack variable λ is used over net spending (as in Eq. (8.10) below), the objective is the sum $\sum_{n=0}^{N_n-1} g_n/\gamma_n$, maximized by setting

$$c[q_g(n)] = -1/\gamma_n, \quad (8.7)$$

$\forall n \in \{0, \dots, N_n - 1\}$, and 0 otherwise. This sum-of-net-spending formulation is the one implemented in Owl.

Variable profile with slack

When the slack variable λ is used (Eq. (5.27) or Eq. (8.10) below), the objective function is the sum of net spending over the full duration of the plan in today's dollars, $\sum_{n=0}^{N_n-1} g_n/\gamma_n$. The coefficient vector is

$$c[q_g(n)] = -1/\gamma_n, \quad (8.8)$$

$\forall n \in \{0, \dots, N_n - 1\}$ and 0 otherwise. In that case, constraint equality Eq. (5.26) is replaced by an inequality. Instead of obeying

$$g_n \xi_0 - g_0 \bar{\xi}_n = 0, \quad (8.9)$$

we impose the inequality constraint

$$(1 - \lambda)g_0 \bar{\xi}_n / \xi_0 \leq g_n \leq (1 + \lambda)g_0 \bar{\xi}_n / \xi_0, \quad (8.10)$$

where λ is the fractional tolerance by which annual net spending is allowed to deviate from the desired profile. It is set via the `spendingSlack` option (an integer percentage, e.g. `spendingSlack=10` gives $\lambda = 0.10$); the default is $\lambda = 0$ (strict profile). It should be noticed that when $\lambda = 0$ the two last equations are equivalent.

8.2 Maximizing bequest

If, on the other hand, one would like to maximize the bequest under the constraint of a desired net spending g_0 , specified for the first year, one would add the following row to $A_e y = v$

$$\begin{aligned} A_e[I(0), q_g(0)] &= 1, \\ v[I(0)] &= g_0, \end{aligned} \tag{8.11}$$

subject to the net spending g_n obeying Eq. (5.26) over time.

The objective function would then be derived from Eq. (8.1) as minimizing the inner product $c \cdot y$, where c is

$$c[q_b(i, j, N_n)] = -(1 - \nu \delta_{j,1}), \tag{8.12}$$

$$\forall i \in \{0, \dots, N_i - 1\},$$

$$\forall j \in \{0, \dots, N_j - 1\},$$

$$\tag{8.13}$$

and 0 otherwise.

8.3 Lexicographic tie-breaking

The coefficient vector c can be augmented with a small positive weight ε to break ties without changing the primary objective in practice. In addition to the coefficients of the chosen primary objective (max spending or max bequest), the implementation adds the following contributions to c :

$$\begin{aligned} c[q_x(i, n)] &\leftarrow c[q_x(i, n)] + \varepsilon, \\ &\quad \forall i \in \{0, \dots, N_i - 1\}, \forall n \in \{0, \dots, N_n - 1\}, \\ c[q_w(1, j, n)] &\leftarrow c[q_w(1, j, n)] + \varepsilon, \\ &\quad \forall j \in \{0, \dots, N_j - 1\}, \forall n \in \{0, \dots, N_n - 1\}, \quad \text{when } N_i = 2. \end{aligned} \tag{8.14}$$

The first line penalizes Roth conversions x_{in} , so that among solutions with the same primary objective value the solver prefers fewer or smaller conversions (reducing churn). The second line penalizes withdrawals w_{1jn} from the second individual ($i = 1$), so that withdrawals from the first individual ($i = 0$) are preferred when otherwise equivalent. The weight ε is chosen small (e.g. of order 10^{-9} in the same units as y) so that the primary objective dominates; it is often enabled by default when optimizing Medicare (IRMAA) to stabilize bracket selection.

9. Rate models for return scenarios

The optimization model described in the preceding chapters requires a sequence $\{\tau_{kn}\}_{n=0}^{N_n-1}$ of annual returns for each asset class k over the plan horizon. Owl provides several models for generating this sequence, ranging from simple fixed-rate assumptions to fully stochastic simulation. This chapter describes the models available and the statistical algorithms underlying the stochastic ones.

9.1 Historical data

All built-in models draw on an annual dataset covering the period from $\mathcal{Y}_{\min} = 1928$ through the most recently completed calendar year $\mathcal{Y}_{\max}$, containing $N_y = \mathcal{Y}_{\max} - \mathcal{Y}_{\min} + 1$ annual observations [19]. Four asset-class return time series are included, corresponding to the index k :

- $k = 0$ S&P 500 total return, including dividends.
- $k = 1$ Moody's Baa corporate bond total return (investment-grade bonds with a moderate default premium).
- $k = 2$ 10-year U.S. Treasury note total return.
- $k = 3$ Consumer Price Index (CPI) inflation. This series serves a dual role: it drives the cumulative inflation factor γ_n (see Chapter 2), and, as the *cash asset*, it proxies a Treasury Inflation-Protected Security (TIPS) with a real yield of zero so that holding cash preserves but does not grow purchasing power.

All values are stored in percent (e.g. 7.0 for a 7% annual return). A user-selected sub-range $[\mathcal{Y}_{\text{frm}}, \mathcal{Y}_{\text{to}}] \subseteq [\mathcal{Y}_{\min}, \mathcal{Y}_{\max}]$ is used by the `historical`, `historical_average`, `historical_gaussian`, `historical_lognormal`, `historical_bootstrap`, `historical_copula`, `vector_ar`, `garch_dcc`, `gmm`, and `hmm` models. We denote the window width

by $N_w = \mathcal{Y}_{\text{to}} - \mathcal{Y}_{\text{frm}} + 1$, and the observed return of asset class k in calendar year y by r_{ky} .

Sequence transformations: reverse and roll. Two optional transforms apply to the extracted sequence before it is used. The *reverse* option reflects the sequence in time, $\tau_{kn} \leftarrow r_{k, \mathcal{Y}_{\text{to}} - n}$, so that the earliest calendar year in the window maps to plan year $N_n - 1$ and the latest to plan year 0. The *roll* option shifts the sequence by s positions, $\tau_{kn} \leftarrow r_{k, \mathcal{Y}_{\text{frm}} + ((n+s) \bmod N_w)}$, with positive s advancing toward later years (and wrapping). Both transforms preserve the set of returns in the window while altering the order; they are useful for exploring sequence-of-returns sensitivity without changing the underlying data. Reverse and roll are supported by all methods that use a historical window (`historical`, `historical_gaussian`, `historical_lognormal`, `historical_bootstrap`, `historical_copula`, `vector_ar`, `garch_dcc`, `gmm`, and `hmm`). Constant-rate methods ignore them.

9.2 Constant rate methods

The simplest approach assigns a constant return to every year of the plan, $\tau_{kn} = \tau_k$ for all n . Four variants are supported:

`trailing_30`

Trailing 30-year geometric mean of annual returns, updated annually.

`conservative and optimistic`

Preset rates based on long-term expert forecasts.

`historical_average`

The geometric mean over the selected historical window,

$$\tau_k = \exp\left(\frac{1}{N_w} \sum_{y=\mathcal{Y}_{\text{frm}}}^{\mathcal{Y}_{\text{to}}} \ln(1 + r_{ky})\right) - 1. \quad (9.1)$$

`user`

Freely specified by the user for each asset class.

9.3 Deterministic varying rates: historical

The `historical` method assigns recorded returns directly to each year of the plan:

$$\tau_{kn} = r_{k, \mathcal{Y}_{\text{frm}} + n}, \quad n = 0, \dots, N_n - 1. \quad (9.2)$$

If the plan horizon N_n exceeds the window width N_w , the sequence is repeated cyclically. The reverse and roll transforms (see §9.1) are applied before the plan runs.

9.4 Stochastic models

Stochastic methods draw N_n annual return vectors $\boldsymbol{\tau}_n = (\tau_{0n}, \tau_{1n}, \tau_{2n}, \tau_{3n})^\top \in \mathbb{R}^{N_k}$ from a probability distribution calibrated on historical data. Because random number generation is involved, a seed can be fixed to obtain reproducible results across runs. Each fresh draw of N_n vectors constitutes one scenario; running many independent scenarios yields a Monte Carlo distribution of outcomes.

9.4.1 Inflation skewness correction

Historical US inflation rates are substantially right-skewed: the distribution has a long right tail from high-inflation episodes (e.g. the 1970s) while low-inflation and deflation years cluster near the median. This asymmetry violates the Gaussian residual assumption shared by four of Owl’s parametric stochastic models: `historical\_gaussian`, `historical\_lognormal`, `vector\_ar`, and `garch\_dcc`.

To correct for this, Owl applies a piecewise-linear (PWL) normalization transform φ to the inflation dimension ($k = 3$) before fitting each of these models, and its inverse φ^{-1} to the generated inflation samples after drawing:

$$\varphi(z; \kappa, s^-, s^+) = \begin{cases} (z - \kappa) s^- + \kappa & z \leq \kappa, \\ (z - \kappa) s^+ + \kappa & z > \kappa, \end{cases} \quad (9.3)$$

where κ is the empirical median of the inflation values in the selected historical window and $s^- > 0$, $s^+ > 0$ are the slopes below and above the median respectively. With $s^- > 1$ and $s^+ < 1$, the transform stretches the short left tail and compresses the long right tail, producing a distribution closer to Gaussian. The inverse is

$$\varphi^{-1}(w; \kappa, s^-, s^+) = \begin{cases} (w - \kappa)/s^- + \kappa & w \leq \kappa, \\ (w - \kappa)/s^+ + \kappa & w > \kappa. \end{cases} \quad (9.4)$$

Automatic calibration The slopes s^- and s^+ are fitted automatically from the same historical window used to calibrate the rate model, so they adapt when the user selects a different date range. They minimize the squared skewness of $\varphi(z)$ subject

to a small regularization toward the identity ($s^- = s^+ = 1$) that ensures a unique, low-distortion solution:

$$(s^-, s^+) = \arg \min_{s^- > 0, s^+ > 0} \left[\text{skew}(\varphi(z))^2 + 0.1((s^- - 1)^2 + (s^+ - 1)^2) \right]. \quad (9.5)$$

For US inflation data over the full 1928–2025 window the optimizer typically finds $s^- \approx 2.3$ and $s^+ \approx 0.8$. The fitted values of κ , s^- , and s^+ are recorded in the debug log at model initialization.

Scope and domain The transform is applied to the inflation dimension only; equities, bonds, and T-notes are unchanged. User-specified models (`gaussian`, `lognormal`) and data-reuse models (`historical`, `historical\_bootstrap`, `historical\_average`) are unaffected. For `historical\_gaussian`, `vector\_ar`, and `garch\_dcc`, φ is applied in *return space*: historical inflation returns $r_{3,y}$ are mapped to $\varphi(r_{3,y})$ before parameter estimation, and φ^{-1} is applied to generated inflation draws before returning them. For `historical\_lognormal`, φ is applied in *log-return space*: the log-returns $\ell_{3,y} = \ln(1 + r_{3,y})$ are mapped to $\varphi(\ell_{3,y})$ before Gaussian fitting, and φ^{-1} is applied to the generated log-return samples before exponentiation. Working in log-space eliminates the constraint $\varphi(r_{3,y}) > -1$ that would otherwise be required for the logarithm to remain well-defined.

9.4.2 Mean anchoring (`constrain_mean`)

An optional post-generation correction is available for `historical\_gaussian`, `historical\_lognormal`, `historical\_copula`, `garch\_dcc`, `gmm`, and `hmm`. When activated, the arithmetic mean of the generated scenario is shifted to match the historical window mean exactly.

Let $\boldsymbol{\tau}^{(1)}, \dots, \boldsymbol{\tau}^{(N_n)} \in \mathbb{R}^{N_k}$ be the N_n annual return vectors produced by any of the stochastic models above. Define the sample mean over the scenario,

$$\hat{\boldsymbol{\mu}} = \frac{1}{N_n} \sum_{n=1}^{N_n} \boldsymbol{\tau}^{(n)}, \quad (9.6)$$

and the historical arithmetic mean over the selected window,

$$\mu_k^{\text{hist}} = \frac{1}{N_w} \sum_{y=\mathcal{Y}_{\text{frm}}}^{\mathcal{Y}_{\text{to}}} r_{ky}. \quad (9.7)$$

The correction shifts every year’s return vector by the same additive offset:

$$\boldsymbol{\tau}^{(n)} \leftarrow \boldsymbol{\tau}^{(n)} + (\boldsymbol{\mu}^{\text{hist}} - \hat{\boldsymbol{\mu}}), \quad n = 1, \dots, N_n. \quad (9.8)$$

This preserves variances, correlations, skewness, volatility clustering, and all other distributional properties, while ensuring that the scenario’s mean return matches the historical reference by construction.

The practical motivation is sampling bias in short scenarios: for small N_n , the sample mean $\hat{\boldsymbol{\mu}}$ can deviate substantially from $\boldsymbol{\mu}^{\text{hist}}$, introducing a systematic optimistic or pessimistic tilt that is purely a finite-sample artefact. Mean anchoring removes this tilt without distorting the shape of the distribution. Return floors ($\tau_{kn} \geq -1$ for $k \in \{0, 1, 2\}$; $\tau_{3n} \geq \tau_{\min}^{\text{infl}}$) are applied *after* the correction.

9.4.3 Multivariate normal: `historical_gaussian` and `gaussian`

Both methods model annual returns as draws from a multivariate normal distribution,

$$\boldsymbol{\tau}_n \stackrel{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad n = 0, \dots, N_n - 1. \quad (9.9)$$

For `historical_gaussian`, the mean vector $\boldsymbol{\mu}$ and the $N_k \times N_k$ covariance matrix $\boldsymbol{\Sigma}$ are estimated from the selected historical window:

$$\mu_k = \frac{1}{N_w} \sum_{y=\mathcal{Y}_{\text{frm}}}^{\mathcal{Y}_{\text{to}}} r_{ky}, \quad (9.10)$$

$$\Sigma_{kk'} = \frac{1}{N_w - 1} \sum_{y=\mathcal{Y}_{\text{frm}}}^{\mathcal{Y}_{\text{to}}} (r_{ky} - \mu_k)(r_{k'y} - \mu_{k'}). \quad (9.11)$$

For `gaussian`, the user supplies $\boldsymbol{\mu}$ directly, together with the standard deviations $\sigma_k = \sqrt{\Sigma_{kk}}$ and the Pearson correlation coefficients $\rho_{kk'} \in [-1, 1]$, from which the covariance matrix is assembled as $\Sigma_{kk'} = \rho_{kk'} \sigma_k \sigma_{k'}$. In both cases, samples are generated as $\boldsymbol{\tau}_n = \boldsymbol{\mu} + L \mathbf{z}_n$, where L is the lower-triangular Cholesky factor of $\boldsymbol{\Sigma}$ and $\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, I)$. By construction, the multivariate normal model treats each year’s returns as independent of all other years; it captures cross-sectional correlation among asset classes but no serial (year-to-year) autocorrelation.

9.4.4 Log-normal models: `lognormal` and `historical_lognormal`

Geometric Brownian Motion (GBM) [20] — the foundation of most option-pricing theory — implies that the price relative $1 + \tau_{k,n}$ is log-normally distributed. The

log-normal model enforces $\tau_{k,n} > -1$ by construction (no total-loss artifacts are possible), produces a right-skewed marginal distribution (large gains are more likely than a symmetric normal would suggest), and is consistent with continuous-time finance theory.

Parameter conversion Given the arithmetic mean m_k and arithmetic standard deviation s_k of asset-class k (both in decimal), the corresponding log-space parameters are [20]

$$\sigma_{Z,k}^2 = \ln \left(1 + \left(\frac{s_k}{1 + m_k} \right)^2 \right), \quad (9.12)$$

$$\mu_{Z,k} = \ln(1 + m_k) - \frac{1}{2} \sigma_{Z,k}^2. \quad (9.13)$$

One can verify that $\mathbb{E}[\exp(Z_k)] - 1 = m_k$ and $\text{Var}[\exp(Z_k) - 1] = s_k^2$ when $Z_k \sim \mathcal{N}(\mu_{Z,k}, \sigma_{Z,k}^2)$.

Sampling A log-space covariance matrix is assembled from the log-space standard deviations $\sigma_{Z,k} = \sqrt{\sigma_{Z,k}^2}$ and the Pearson correlation matrix $\mathbf{C}$ with entries $\rho_{kk'}$,

$$(\boldsymbol{\Sigma}_Z)_{kk'} = \rho_{kk'} \sigma_{Z,k} \sigma_{Z,k'}. \quad (9.14)$$

Then N_n vectors of log-returns are drawn jointly,

$$\mathbf{Z}_n \stackrel{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\mu}_Z, \boldsymbol{\Sigma}_Z), \quad n = 0, \dots, N_n - 1, \quad (9.15)$$

and annual returns are recovered by exponentiation,

$$\tau_{k,n} = \exp(Z_{k,n}) - 1. \quad (9.16)$$

Applying the correlation matrix directly in log-space is a standard approximation, exact for jointly log-normal variates and accurate for moderate return magnitudes.

lognormal — user-supplied parameters The user provides arithmetic means m_k and standard deviations s_k (in percent) together with Pearson correlations $\rho_{kk'}$. These are converted to log-space via Eqs. (9.12)–(9.13) before sampling.

historical_lognormal — history-calibrated parameters Parameters are estimated directly in log-space from the selected historical window without any intermediate conversion to arithmetic parameters. Let $\ell_{ky} = \ln(1 + r_{ky})$ be the log-return of asset class k in calendar year y . Then

$$\mu_{Z,k} = \frac{1}{N_w} \sum_{y=\mathcal{Y}_{\text{frm}}}^{\mathcal{Y}_{\text{to}}} \ell_{ky}, \quad (9.17)$$

$$(\Sigma_Z)_{kk'} = \frac{1}{N_w - 1} \sum_{y=\mathcal{Y}_{\text{frm}}}^{\mathcal{Y}_{\text{to}}} (\ell_{ky} - \mu_{Z,k})(\ell_{k'y} - \mu_{Z,k'}). \quad (9.18)$$

Arithmetic statistics displayed in the UI are recovered from the log-space moments via the inverse of Eqs. (9.12)–(9.13):

$$m_k = \exp(\mu_{Z,k} + \frac{1}{2}(\Sigma_Z)_{kk}) - 1, \quad (9.19)$$

$$s_k = (1 + m_k) \sqrt{\exp((\Sigma_Z)_{kk}) - 1}. \quad (9.20)$$

9.4.5 Bootstrap sequence-of-returns: `historical_bootstrap`

The bootstrap sequence-of-returns model resamples N_n annual return vectors directly from the observed window $\{\mathbf{r}_y\}_{y=\mathcal{Y}_{\text{frm}}}^{\mathcal{Y}_{\text{to}}}$, where $\mathbf{r}_y = (r_{0,y}, \dots, r_{N_k-1,y})^\top$. Resampling from actual data preserves the empirical marginal distributions exactly, including skewness, fat tails, and any departure from normality. Four resampling strategies are supported.

IID bootstrap Each year is drawn independently and with replacement from the full window [21]:

$$\boldsymbol{\tau}_n = \mathbf{r}_{y_n}, \quad y_n \stackrel{\text{iid}}{\sim} \text{Uniform}\{\mathcal{Y}_{\text{frm}}, \dots, \mathcal{Y}_{\text{to}}\}. \quad (9.21)$$

This is the simplest and default variant; it destroys all serial correlation present in the original data.

Block bootstrap To preserve short-run serial correlation, the block bootstrap of Künsch [22] draws $\lceil N_n/b \rceil$ starting positions s_ℓ uniformly from $\{\mathcal{Y}_{\text{frm}}, \dots, \mathcal{Y}_{\text{to}} - b + 1\}$ and concatenates the corresponding blocks of b consecutive years, truncating to N_n years:

$$(\boldsymbol{\tau}_0, \dots, \boldsymbol{\tau}_{N_n-1}) = (\mathbf{r}_{s_1}, \dots, \mathbf{r}_{s_1+b-1}, \mathbf{r}_{s_2}, \dots, \mathbf{r}_{s_2+b-1}, \dots). \quad (9.22)$$

The parameter b is the block size, chosen by the user.

Circular block bootstrap A variant due to Politis and Romano [23] treats the historical window as circular, wrapping from $\mathcal{Y}_{\text{to}}$ back to $\mathcal{Y}_{\text{frm}}$. Starting positions are drawn uniformly from the full window $\{\mathcal{Y}_{\text{frm}}, \dots, \mathcal{Y}_{\text{to}}\}$, and block membership wraps around modulo N_w . This removes the boundary effect present in the block bootstrap, where observations near the endpoints of the window are underrepresented.

Stationary bootstrap In the stationary bootstrap of Politis and Romano [24], block lengths are random rather than fixed. Each block length L_ℓ is drawn from a geometric distribution,

$$P(L_\ell = \ell) = p_b (1 - p_b)^{\ell-1}, \quad \ell = 1, 2, \dots, \quad (9.23)$$

with parameter $p_b = 1/b$, so that the expected block length equals b . This choice produces a strictly stationary bootstrap distribution, resolving the boundary discontinuities of the fixed-block method while still approximately preserving serial correlation at lags up to order b .

9.4.6 Vector autoregression: `vector_ar`

The VAR(1) model [25, 26] extends the multivariate normal approach by explicitly capturing year-to-year serial correlations in asset returns, such as momentum and mean-reversion. The generating model is

$$\mathbf{y}_t = \mathbf{c} + \mathbf{A}\mathbf{y}_{t-1} + \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \quad (9.24)$$

where $\mathbf{y}_t \in \mathbb{R}^{N_k}$ is the vector of asset-class returns at time t , $\mathbf{c} \in \mathbb{R}^{N_k}$ is a constant intercept vector, $\mathbf{A} \in \mathbb{R}^{N_k \times N_k}$ is the transition matrix encoding one-period predictability, and $\boldsymbol{\Sigma}$ is the contemporaneous residual covariance matrix. For $\mathbf{A} = \mathbf{0}$, Eq. (9.24) reduces to the multivariate normal model of the `historical_gaussian` method.

Parameter estimation Given N_w historical observations $\mathbf{y}_1, \dots, \mathbf{y}_{N_w}$, parameters $\mathbf{c}$ and $\mathbf{A}$ are estimated jointly by ordinary least squares (OLS). Stacking observations into matrices

$$Y = [\mathbf{y}_2 \mid \dots \mid \mathbf{y}_{N_w}]^\top \in \mathbb{R}^{(N_w-1) \times N_k}, \quad (9.25)$$

$$X = [\mathbf{1} \mid \mathbf{y}_1 \mid \dots \mid \mathbf{y}_{N_w-1}]^\top \in \mathbb{R}^{(N_w-1) \times (N_k+1)}, \quad (9.26)$$

the OLS solution $B = (X^\top X)^{-1} X^\top Y$ yields the intercept $\mathbf{c} = B[0, :]$ and transition matrix $\mathbf{A} = B[1 :, :]^\top$. The residual covariance is estimated as

$$\boldsymbol{\Sigma} = \frac{(Y - XB)^\top (Y - XB)}{N_w - N_k - 1}. \quad (9.27)$$

Stationarity and spectral shrinkage Stationarity of the VAR(1) process requires that all eigenvalues of $\mathbf{A}$ have modulus strictly less than one. Let $\rho(\mathbf{A}) = \max_i |\lambda_i(\mathbf{A})|$ denote the spectral radius. If $\rho(\mathbf{A}) \geq 0.95$ (near or above the unit circle), the transition matrix is rescaled as

$$\mathbf{A} \leftarrow \frac{0.95}{\rho(\mathbf{A})} \mathbf{A}, \quad (9.28)$$

ensuring $\rho(\mathbf{A}) = 0.95 < 1$ while preserving the direction of every eigenmode. This spectral shrinkage step can be disabled by the user, at the cost of potentially non-stationary sequences.

Sequence generation Starting from $\mathbf{y}_0 = \boldsymbol{\mu}$ (the historical sample mean), sequences are generated recursively from Eq. (9.24) using $\boldsymbol{\varepsilon}_t = L\mathbf{z}_t$, where L is the Cholesky factor of $\boldsymbol{\Sigma}$ and $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, I)$. The VAR(1) model produces synthetic sequences of arbitrary length while capturing both the cross-sectional covariance structure and the serial autocorrelation of asset returns.

9.4.7 DCC-GARCH(1,1): garch_dcc

The DCC-GARCH(1,1) model [27, 28] extends the multivariate normal approach by allowing both the per-asset conditional variance and the cross-asset correlation matrix to vary over time. It is fitted in two steps on the selected historical window.

Step 1 — per-asset GARCH(1,1) Let $\varepsilon_{kt} = y_{k,t} - \mu_k$ be the demeaned return of asset class k at year t . The conditional variance h_{kt} evolves as

$$h_{kt} = \omega_k + \alpha_k \varepsilon_{k,t-1}^2 + \beta_k h_{k,t-1}, \quad (9.29)$$

with $h_{k,0} = \text{Var}(\varepsilon_k)$. The parameters $(\omega_k, \alpha_k, \beta_k)$ are estimated independently for each asset by maximizing the Gaussian log-likelihood

$$\ell_k = -\frac{1}{2} \sum_{t=1}^{N_w} [\ln h_{kt} + \varepsilon_{kt}^2 / h_{kt}], \quad (9.30)$$

subject to $\omega_k > 0$, $\alpha_k, \beta_k > 0$, and the covariance-stationarity constraint $\alpha_k + \beta_k < 1$. Optimization uses L-BFGS-B; if convergence fails, an EWMA fallback ($\alpha_k = 0.06$, $\beta_k = 0.94$) is used instead.

The standardized residuals $z_{kt} = \varepsilon_{kt} / \sqrt{h_{kt}}$ are passed to the second step.

Step 2 — DCC dynamics Let $\bar{\mathbf{Q}} = \frac{1}{N_w} \mathbf{Z}^\top \mathbf{Z}$ be the unconditional covariance of $\mathbf{Z} = [z_{kt}]$. The quasi-correlation matrix $\mathbf{Q}_t$ evolves as

$$\mathbf{Q}_t = (1 - a - b) \bar{\mathbf{Q}} + a \mathbf{z}_{t-1} \mathbf{z}_{t-1}^\top + b \mathbf{Q}_{t-1}, \quad (9.31)$$

with $a, b > 0$ and $a + b < 1$. The time-varying correlation matrix is obtained by normalizing:

$$\mathbf{R}_t = \text{diag}(\mathbf{Q}_t)^{-1/2} \mathbf{Q}_t \text{diag}(\mathbf{Q}_t)^{-1/2}. \quad (9.32)$$

The DCC parameters (a, b) are estimated by maximizing the concentrated log-likelihood

$$\ell_{\text{DCC}} = -\frac{1}{2} \sum_{t=1}^{N_w} [\ln |\mathbf{R}_t| + \mathbf{z}_t^\top \mathbf{R}_t^{-1} \mathbf{z}_t]. \quad (9.33)$$

Sequence generation Warm-start states $\mathbf{Q}_0$ and $\mathbf{h}_0 = (h_{1,N_w}, \dots, h_{N_k,N_w})$ are obtained from a forward pass through the historical data. Each simulated year is then drawn as

$$\mathbf{y}_t = \boldsymbol{\mu} + \text{diag}(\mathbf{h}_t)^{1/2} L(\mathbf{R}_t) \mathbf{z}_t, \quad \mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, I), \quad (9.34)$$

where $L(\mathbf{R}_t)$ is the lower Cholesky factor of $\mathbf{R}_t$, followed by GARCH and DCC updates via Eqs. (9.29) and (9.31). This produces sequences that exhibit *volatility clustering* — large shocks in one year increase conditional variance in the next — and *time-varying cross-asset correlations* that spike during simulated market-stress episodes, consistent with empirical evidence in financial return data.

9.4.8 Gaussian mixture model: gmm

The Gaussian mixture model (GMM) [29] generalizes the single multivariate normal of `historical\_gaussian` by representing the joint return distribution as a weighted sum of K multivariate Gaussian components:

$$p(\mathbf{r}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad \pi_k \geq 0, \quad \sum_{k=1}^K \pi_k = 1. \quad (9.35)$$

Each component k has its own mean vector $\boldsymbol{\mu}_k \in \mathbb{R}^{N_k}$, full covariance matrix $\boldsymbol{\Sigma}_k \in \mathbb{R}^{N_k \times N_k}$, and mixture weight π_k .

Motivation Historical return data are not well-described by a single Gaussian. The empirical distribution exhibits fat tails, negative skewness for equities, and distinct clustering consistent with alternating market regimes — a prolonged bull market, a bear market, and acute crisis periods. A GMM with $K = 3$ components can capture these regimes jointly across all asset classes, so that in a “crisis” component, both equities and investment-grade bonds may fall together (correlation breakdown), while in a “bull” component the usual negative stock–bond correlation is restored. Each component’s Σ_k encodes the cross-asset covariance *conditional on being in that regime*.

Fitting The parameters $\{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1}^K$ are estimated from the N_w annual return vectors in the selected historical window $[\mathcal{Y}_{\text{frm}}, \mathcal{Y}_{\text{to}}]$ by Expectation-Maximization (EM) [30]. The EM algorithm alternates between:

- **E-step:** compute the posterior responsibility $\gamma_{nk} = \pi_k \mathcal{N}(\mathbf{r}_n; \boldsymbol{\mu}_k, \Sigma_k) / \sum_j \pi_j \mathcal{N}(\mathbf{r}_n; \boldsymbol{\mu}_j, \Sigma_j)$ for each observation n and component k ;
- **M-step:** update $\pi_k = \frac{1}{N_w} \sum_n \gamma_{nk}$, $\boldsymbol{\mu}_k = \frac{\sum_n \gamma_{nk} \mathbf{r}_n}{\sum_n \gamma_{nk}}$, $\Sigma_k = \frac{\sum_n \gamma_{nk} (\mathbf{r}_n - \boldsymbol{\mu}_k)(\mathbf{r}_n - \boldsymbol{\mu}_k)^\top}{\sum_n \gamma_{nk}}$.

Convergence is declared when the change in log-likelihood falls below a tolerance threshold.

Sequence generation To draw one simulated year of returns:

1. Sample a regime index $k^* \sim \text{Categorical}(\boldsymbol{\pi})$.
2. Draw $\boldsymbol{\tau}_n \sim \mathcal{N}(\boldsymbol{\mu}_{k^*}, \Sigma_{k^*})$.

Each of the N_n plan years is drawn independently, so the model captures cross-sectional correlation within each regime but no serial autocorrelation across years. The number of components K is a user parameter (default $K = 3$); the covariance structure can also be restricted to tied, diagonal, or spherical forms.

9.4.9 Hidden Markov model: hmm

The hidden Markov model (HMM) [31] extends the GMM by adding temporal structure to the regime sequence. An unobserved discrete state $s_n \in \{1, \dots, K\}$ evolves as a first-order Markov chain with transition matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$, $A_{kj} = P(s_{n+1} = j \mid$

$s_n = k$), with $\sum_j A_{kj} = 1$, and each year's returns are drawn from the multivariate Gaussian of the current state:

$$s_{n+1} \mid s_n = k \sim \text{Categorical}(\mathbf{A}_k), \quad \boldsymbol{\tau}_n \mid s_n = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \quad (9.36)$$

The emission parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ have the same interpretation as in the GMM; the key new quantity is $\mathbf{A}$.

Motivation The GMM treats each simulated year as an independent regime draw. Historical return data exhibit regime *persistence*: bull and bear markets tend to continue for several years before switching. The HMM encodes this by coupling consecutive years through $\mathbf{A}$, so a diagonal entry A_{kk} close to 1 produces multi-year runs of the same regime. This is directly relevant to sequence-of-returns risk: a retiree entering a bear-market regime faces an elevated probability of further down years before the regime changes, amplifying the compounding impact of early losses in ways the i.i.d. GMM cannot reproduce.

Fitting Given N_w historical return vectors $\mathbf{r}_0, \dots, \mathbf{r}_{N_w-1}$, the parameters $\{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ are estimated by the Baum-Welch algorithm [33], which applies EM [30] to the HMM likelihood. Each iteration runs a scaled forward-backward pass and a parameter update.

Forward pass:

$$\hat{\alpha}_0(k) = \pi_k b_k(\mathbf{r}_0), \quad c_0 = \sum_k \hat{\alpha}_0(k), \quad \hat{\alpha}_0 \leftarrow \hat{\alpha}_0 / c_0, \quad (9.37)$$

$$\hat{\alpha}_n(j) = b_j(\mathbf{r}_n) \sum_k \hat{\alpha}_{n-1}(k) A_{kj}, \quad c_n = \sum_j \hat{\alpha}_n(j), \quad \hat{\alpha}_n \leftarrow \hat{\alpha}_n / c_n, \quad (9.38)$$

where $b_k(\mathbf{r}) = \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. The log-likelihood is $\ell = \sum_n \log c_n$.

Backward pass:

$$\hat{\beta}_{N_w-1}(k) = 1, \quad \hat{\beta}_n(k) = \frac{1}{c_{n+1}} \sum_j A_{kj} b_j(\mathbf{r}_{n+1}) \hat{\beta}_{n+1}(j). \quad (9.39)$$

Posterior quantities:

$$\gamma_n(k) = \hat{\alpha}_n(k) \hat{\beta}_n(k) / \sum_j \hat{\alpha}_n(j) \hat{\beta}_n(j), \quad \xi_n(k, j) \propto \hat{\alpha}_n(k) A_{kj} b_j(\mathbf{r}_{n+1}) \hat{\beta}_{n+1}(j). \quad (9.40)$$

M-step:

$$\pi_k = \gamma_0(k), \quad (9.41)$$

$$A_{kj} = \frac{\sum_{n=0}^{N_w-2} \xi_n(k, j) + \delta}{\sum_{n=0}^{N_w-2} \gamma_n(k) + K\delta}, \quad (9.42)$$

$$\boldsymbol{\mu}_k = \frac{\sum_n \gamma_n(k) \mathbf{r}_n}{\sum_n \gamma_n(k)}, \quad \boldsymbol{\Sigma}_k = \frac{\sum_n \gamma_n(k) (\mathbf{r}_n - \boldsymbol{\mu}_k)(\mathbf{r}_n - \boldsymbol{\mu}_k)^\top}{\sum_n \gamma_n(k)} + \varepsilon \mathbf{I}, \quad (9.43)$$

where $\delta > 0$ (default 10^{-3}) is a Laplace smoothing term that prevents zero-probability transitions for regimes not visited in the historical window, and $\varepsilon = 10^{-6}$ regularizes the covariances. Convergence is declared when the improvement in ℓ falls below a tolerance; the transition matrix is initialized to an 80% self-transition prior, $A_{kk} = 0.8$, to encourage persistence from the first iteration.

Sequence generation Let $\boldsymbol{\pi}_\infty$ denote the stationary distribution satisfying $\boldsymbol{\pi}_\infty = \boldsymbol{\pi}_\infty \mathbf{A}$, computed by power iteration. A plan of N_n years is simulated as:

1. Draw the initial state $s_0 \sim \boldsymbol{\pi}_\infty$ (or from a user-specified `init_regime`).
2. For $n = 0, 1, \dots, N_n - 1$: draw $\boldsymbol{\tau}_n \sim \mathcal{N}(\boldsymbol{\mu}_{s_n}, \boldsymbol{\Sigma}_{s_n})$, then transition $s_{n+1} \sim \text{Categorical}(\mathbf{A}_{s_n, \cdot})$.

Fitted on 1928–2025 annual US data with $K = 3$, the diagonal entries A_{kk} typically exceed 0.6, yielding mean regime-run lengths of two to four years. The HMM is strictly more expressive than the GMM: if $\mathbf{A}$ is set to the outer product $\mathbf{1}\boldsymbol{\pi}^\top$ (all rows identical), the model reduces to the i.i.d. GMM, and the HMM log-likelihood is always at least as large.

9.4.10 Gaussian copula: `historical_copula`

The Gaussian copula model [32] generates synthetic return sequences whose marginal distributions match the historical data exactly, while their joint dependence structure is captured by a parametric Gaussian copula. Unlike `historical\_gaussian` and `historical\_lognormal`, which impose a Gaussian (or log-normal) shape on every marginal, this method is fully non-parametric in the marginals: it preserves the exact empirical distribution of each asset class — including the left skew of equity returns, the right skew of T-Note returns, and the right skew of inflation.

Motivation Sklar’s theorem [32] states that any joint distribution can be decomposed into its marginal distributions and a copula that captures all dependence. In the Gaussian copula family, the dependence is parameterized by a $N_k \times N_k$ correlation matrix $\mathbf{R}$, separating the fitting of individual-asset behavior from the fitting of cross-asset correlations. For retirement planning, this matters because inflation in particular is neither Gaussian nor log-normal; preserving its empirical shape avoids systematic bias in real spending projections.

Fitting Let $\mathbf{r}_y = (r_{1y}, \dots, r_{N_k, y})^\top$ denote the observed return vector for calendar year y in the selected window $[\mathcal{Y}_{\text{frm}}, \mathcal{Y}_{\text{to}}]$, with N_w total observations.

Step 1 — rank-based empirical CDF. For each asset class k , map the N_w observations to the uniform grid via rank statistics:

$$U_{ky} = \frac{\text{rank}(r_{ky}) - \frac{1}{2}}{N_w}, \quad y = \mathcal{Y}_{\text{frm}}, \dots, \mathcal{Y}_{\text{to}}, \quad (9.44)$$

where rank is computed within the window for each k separately. The $-\frac{1}{2}$ centering keeps all values strictly inside $(0, 1)$ so that the next step is well-defined.

Step 2 — normal scores. Apply the standard normal quantile function to obtain copula variates:

$$Z_{ky} = \Phi^{-1}(U_{ky}), \quad (9.45)$$

where Φ denotes the standard normal CDF.

Step 3 — copula correlation matrix. Estimate the $N_k \times N_k$ Gaussian copula correlation matrix from the normal scores:

$$\mathbf{R} = \text{corrcoef}(\mathbf{Z}^\top), \quad (9.46)$$

where $\mathbf{Z} \in \mathbb{R}^{N_w \times N_k}$ collects all normal-score rows. $\mathbf{R}$ captures Pearson (linear) correlation in the normal-score space, which corresponds to Spearman rank correlation in the original return space.

Sequence generation To draw one plan year of synthetic returns:

Step 4 — sample from the copula. Draw a standard multivariate normal vector:

$$\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \quad (9.47)$$

Step 5 — map back to the original scale. Convert to uniform scores and apply the empirical quantile function for each asset class:

$$u_{kn} = \Phi(z_{kn}), \quad \tau_{kn} = \hat{Q}_k(u_{kn}), \quad (9.48)$$

where $\hat{Q}_k$ is the empirical quantile function of asset k , obtained by linear interpolation over the N_w sorted historical values $r_{k,(1)} \leq \dots \leq r_{k,(N_w)}$ at the grid $\{(i - \frac{1}{2})/N_w\}_{i=1}^{N_w}$. Values of u_{kn} outside $[u_{k,1}, u_{k,N_w}]$ are clamped to the observed $[\min, \max]$, preventing extrapolation beyond the historical record. The resolution of $\hat{Q}_k$ equals N_w , so a wider window provides finer quantile resolution.

Step 6 — inflation floor. To exclude the Great Depression artefact of deeply negative inflation, the generated inflation series is floored at -5% .

Each of the N_n plan years is drawn independently; the model captures cross-asset rank correlation but no serial autocorrelation across years.

9.5 User-supplied rates: dataframe

The `dataframe` method allows the user to bypass all built-in rate models and supply the return matrix $\{\tau_{kn}\}$ directly as a pandas `DataFrame` (one column per asset class, one row per plan year). This is useful for custom scenarios, external forecasts, or deterministic stress tests not expressible through the other methods. When the `DataFrame` has fewer rows than the plan horizon N_n , the sequence is repeated cyclically, matching the behavior of the `historical` method. No statistical estimation or sampling is performed; the user bears full responsibility for the consistency and units of the supplied data. This method is accessed programmatically via the Python API (`setRates("dataframe", ...)`); it is not available through the TOML configuration file.

9.6 Monte Carlo and stress testing

In Monte Carlo mode, Owl calls one of the stochastic models (`historical_gaussian`, `historical_lognormal`, `gaussian`, `lognormal`, `historical_bootstrap`, `historical_copula`, `vector_ar`, `garch_dcc`, `gmm`, or `hmm`) independently N_s times, each time solving the full optimization problem with a different rate scenario. The distribution of resulting optimal spending or bequest values across the N_s scenarios provides a probabilistic assessment of the retirement plan.

In stress-testing mode (Historical Range), the `historical` method is applied repeatedly: the starting year $\mathcal{Y}_{\text{frm}}$ is advanced one year at a time over a specified calendar range, producing one deterministic solve per starting year (by default). This sweep is equivalent to asking: “How would this plan have fared had the retiree begun at every historical starting year?” When the *augmented* option is enabled, the run is expanded: for each starting year, every *(reverse, roll)* pair in $\{\text{false}, \text{true}\} \times$

$\{0, \dots, N_n - 1\}$ is executed, yielding $2N_n$ sequences per year instead of one. For a 30-year plan over 50 years of history, that produces $50 \times 60 = 3000$ outcomes rather than 50, providing a much denser sample for the result histogram while still using only observed historical returns. Reverse and roll (see §9.1) alter the ordering of the same returns; augmented mode systematically explores all such orderings.

Robustness of first-year decisions Only the first year’s decisions of a plan are executed; later years are re-optimized as returns realize. Owl therefore also reports, across the same scenario set, the distribution of the *first-year* decisions: per-person Roth conversion percentiles with the fraction of scenarios recommending a conversion and an agreement metric (share of scenarios within 10% of the median), withdrawals by account, net spending, and the modal top ordinary-income bracket (`summarize_year1` in `stresstests.py`, surfaced through the `run_year1_robustness` assistant tool). A statement such as “convert \$45k–\$60k this year in 87% of scenarios” is more decision-relevant than any single-scenario schedule. Each scenario is solved with perfect foresight of its own return path, so the spread *understates* true decision uncertainty; a two-stage stochastic formulation committing the first-year decisions across scenarios — the natural extension on top of the existing Benders infrastructure of Sec. (3.9) — would price that uncertainty exactly and is left as future work.

10. Stochastic Spending Optimization and Longevity Risk

A natural measure of plan robustness is the *Probability of Success* (PoS), defined as the fraction of scenarios in which the chosen spending level is fully sustained throughout the plan horizon. While intuitive, PoS is a binary criterion: a plan that falls short by one percent and a plan that collapses completely are counted identically. Two plans with the same PoS may therefore carry very different risk profiles.

To address this limitation, Owl adopts a framework that treats the retirement spending decision as a commitment made *before* the sequence of returns is known. The retiree commits to a single real spending level g^* , and for each scenario s the shortfall—the amount by which g^* exceeds what the scenario can sustain—is measured explicitly. Tracing the trade-off between committed spending and expected shortfall as a two-dimensional Pareto frontier, controlled by a risk-aversion parameter, yields a richer picture of plan risk that is analogous to the mean-variance efficient frontier of Markowitz portfolio theory [34]. The conservative extreme of the frontier recovers the classical safe withdrawal rate of Bengen [35]: the spending level that succeeds in every scenario.

10.1 Scenario bases and the spending commitment

Let $s = 0, 1, \dots, S - 1$ denote the scenario index (0-based, following the C-array convention used throughout this document), where S is the total number of scenarios considered. For each scenario s , Owl solves one full optimization problem (the MILP described in the preceding chapters) with the return sequence $\{\tau_{kn}^{(s)}\}$ of that scenario. The result is the *scenario basis*

$$g_s = \max\{g : \text{the plan is feasible under scenario } s \text{ at spending level } g\}. \quad (10.1)$$

Because g_s is computed by the MILP, it incorporates all tax rules, RMDs, Social Security, Medicare, and Roth conversion constraints exactly.

The retiree must commit to a single first-year real spending level g^* *before* observing which scenario will unfold. In scenario s the *shortfall* is

$$\varsigma_s = \max(0, g^* - g_s). \quad (10.2)$$

Scenario s is a *success* when $\varsigma_s = 0$. The *Probability of Success* is the fraction of scenarios without shortfall:

$$\varrho = \frac{1}{S} \#\{s : \varsigma_s = 0\}, \quad (10.3)$$

and the *mean shortfall* is

$$\bar{\varsigma} = \frac{1}{S} \sum_{s=0}^{S-1} \varsigma_s. \quad (10.4)$$

10.2 The commitment LP and efficient frontier

Given the scenario bases $\{g_s\}$ and a non-negative risk-aversion parameter $\Lambda \geq 0$, the optimal committed spending is determined by the *commitment LP*:

$$\begin{aligned} & \text{maximize} && g^* - \frac{\Lambda}{S} \sum_{s=0}^{S-1} \varsigma_s \\ & \text{subject to} && \varsigma_s \geq g^* - g_s, && s = 0, \dots, S-1, \\ & && \varsigma_s \geq 0, && s = 0, \dots, S-1, \\ & && 0 \leq g^* \leq \max_s g_s. \end{aligned} \quad (10.5)$$

The decision variables are g^* and the S shortfall variables $\{\varsigma_s\}$. The parameter Λ controls the spending–shortfall trade-off:

- $\Lambda = 0$: maximize g^* without any shortfall penalty. This is the most aggressive point on the frontier; it corresponds to committing to the best-case scenario basis, $g^* = \max_s g_s$.
- $\Lambda \rightarrow \infty$: the shortfall penalty dominates, forcing $\varsigma_s = 0$ for all s . This is the most conservative point; it recovers the safe withdrawal rate $g^* = \min_s g_s$.

The *efficient frontier* is the curve traced in the $(g^*, \bar{\varsigma})$ plane as Λ is swept over a logarithmic grid of 61 values from 0 to 10^3 . Every frontier point is Pareto-optimal: no other committed spending level achieves simultaneously higher g^* and lower $\bar{\varsigma}$. Table 10.1 draws the analogy between this framework and mean-variance portfolio theory.

Concept	Portfolio theory	Spending framework
Decision variable	Portfolio weights	Committed spending g^*
Objective	Expected return	Committed spending g^*
Risk measure	Variance σ^2	Mean shortfall $\bar{\varsigma}$
Risk-aversion param	Λ	Λ
Conservative extreme	Min-variance portfolio	Safe withdrawal rate
Aggressive extreme	Max-return portfolio	Best-case scenario

Table 10.1: Analogy between Markowitz mean-variance portfolio theory and the spending/shortfall efficient frontier.

Connection to Conditional Value-at-Risk. The mean shortfall $\bar{\varsigma}$ is related to the Conditional Value-at-Risk (CVaR) [36, 37] at confidence level ϱ by

$$\text{CVaR}_{\varrho} = \frac{\bar{\varsigma}}{1 - \varrho}. \quad (10.6)$$

CVaR is a coherent risk measure that quantifies the expected shortfall conditional on being in the failure region. Because the commitment LP (10.5) directly minimizes mean shortfall, it is equivalent to minimizing CVaR at the corresponding success rate.

10.3 Longevity risk

All scenario analyses described so far use a fixed planning horizon N_n equal to the assumed life expectancy. In practice, however, the time of death is uncertain, and the risk of outliving one’s assets—*longevity risk*—is a central concern in retirement planning. Owl models longevity risk by randomizing the planning horizon within each Monte Carlo scenario.

For each Monte Carlo scenario s and each individual $i \in \{0, \dots, N_i - 1\}$, an age-at-death $T_i^{(s)}$ is drawn independently from the SSA period life table conditioned on having survived to the individual’s current age:

$$T_i^{(s)} \sim F_i(\cdot \mid \text{current age}_i, \text{sex}_i), \quad i = 0, \dots, N_i - 1, \quad (10.7)$$

where F_i is the discrete conditional survival distribution. For a couple, the plan must remain funded until the last survivor, so the effective plan horizon for scenario s is

$$T^{(s)} = \max_i T_i^{(s)}. \quad (10.8)$$

Each scenario is solved with a fresh plan clone whose life expectancy is set to the drawn last-survivor horizon $T^{(s)}$, so the MILP horizon matches the sampled mortality outcome exactly. Scenarios with $T^{(s)} \leq 1$ year are pre-assigned $g_s = 0$ to handle the degenerate case of an imminent end of horizon.

10.4 Scenario generation

The scenario bases $\{g_s\}$ are generated by one of two methods. The rate models underlying each method are described in Chapter 9.

Historical Each calendar starting year $\mathcal{Y}_{\text{frm}}, \dots, \mathcal{Y}_{\text{to}} - N_n + 1$ yields one fixed real-return sequence and therefore one scenario. S equals the number of valid starting years. Augmented mode expands this set by applying all (*reverse, roll*) combinations, yielding up to $2N_n$ sequences per starting year.

Monte Carlo S independent return paths are drawn from the calibrated multivariate log-normal model (see Chapter 9). The resulting bases span the full tail of the return distribution, including sequences more extreme than any observed historically. Longevity risk sampling is available in this mode only (see below).

10.5 Implementation in Owl

The stress-test computation is implemented in `stresstests.py` and exposed through the `run_stochastic_spending` method of the `Plan` class. Each of the S scenarios is solved independently and in parallel using a `ThreadPoolExecutor` with up to `cpu_count` workers; each worker receives an independent clone of the base plan and calls `plan.solve` with the "maxSpending" objective. Scenarios for which Owl finds no feasible solution are assigned $g_s = 0$, contributing a full shortfall $\varsigma_s = g^*$ to the commitment LP—the most conservative treatment consistent with the framework. Once all S bases are collected, the commitment LP (10.5) is solved for each of the 61 values of Λ ; because the LP has only $S + 1$ variables, this final sweep is fast relative to the MILP scenario solves.

The Streamlit interface (`ui/Spending_Optimization.py`) caches the scenario data; adjusting the target success rate ϱ via the slider merely looks up the corresponding frontier point without re-solving any MILP. An integer seed may be supplied for reproducible Monte Carlo paths and longevity draws. The `run_stochastic_`

`spending` call returns a dictionary with keys `bases`, `frontier\_g`, `frontier\_prob`, `frontier\_shortfall`, `lambdas`, and `drawn\_lifespans`.

Limitations and assumptions

- Longevity risk sampling is supported only with Monte Carlo scenarios. Historical scenarios have a bounded return series; a drawn lifespan could exceed the length of available data, making the combination infeasible.
- Scenario bases are computed independently: the return sequences of different scenarios are assumed to be statistically independent. This is exact for Monte Carlo but approximate for historical scenarios, which share overlapping return sub-sequences when the plan horizon exceeds one year.
- The historical scenario set is limited by the length of the available dataset. For a 30-year plan, only starting years up to $\mathcal{Y}_{\max} - N_n + 1$ are valid, reducing S as the horizon grows.
- The commitment LP assumes that the scenario bases $\{g_s\}$ are fixed parameters independent of g^* . This is valid because Owl's MILP maximizes spending given the return sequence, so g_s does not depend on the committed level.
- Longevity draws use SSA period life tables conditioned on current age. Cohort tables, which account for expected future mortality improvements, are not currently used.

11. Optimization of Social Security Claiming Ages

The age at which an individual claims Social Security retirement benefits is one of the most consequential financial decisions in retirement planning. Claiming early—as young as age 62—provides income sooner but at a permanently reduced rate; deferring to age 70 maximizes the monthly benefit through delayed retirement credits but foregoes years of payments. The optimal age depends on individual longevity expectations, household income, tax situation, and the presence of a spouse whose spousal and survivor benefits are linked to both partners' claiming decisions.

In Owl's base operating mode, the user supplies a fixed claiming age for each individual and the Social Security benefit stream $\bar{\zeta}_{in}$ is computed as a deterministic parameter. When the option `withSSAges="optimize"` is set, the claiming age itself becomes a decision of the MILP: binary variables select one month from the 62-to-70 grid for each individual, a precomputed benefit table translates that selection into an annual income stream, and spousal and survivor adjustments are handled through the self-consistent (SC) iteration described in Chapter 3.

11.1 The claiming-age grid and benefit table

Social Security claiming ages are discretized to a monthly grid. Let $K = 97$ be the number of grid points, indexed $m = 0, 1, \dots, K - 1$, with conventional claiming age

$$\alpha_m = 62 + \frac{m}{12}, \quad m = 0, 1, \dots, 96, \quad (11.1)$$

spanning 62.0 to 70.0 years in monthly increments. Not every grid point is necessarily eligible: individuals born after the first two days of a month may not begin benefits until $\alpha_1 = 62\frac{1}{12}$ due to an SSA birth-date rule.

For each individual i , each claiming-age index m , and each plan year n , Owl

precomputes the own-benefit amount

$$\mathcal{B}_{imn} = 12 \cdot \text{PIA}_i \cdot \Phi(\alpha_m, \text{FRA}_i) \cdot \gamma_n \cdot \varphi_{imn}, \quad (11.2)$$

where PIA_i is the monthly Primary Insurance Amount, Φ is the claiming-age adjustment factor (Section 11.2), γ_n is the cumulative inflation factor in year n , and $\varphi_{imn} \in [0, 1]$ is a partial-year fraction applied in the first payment year to account for the fractional month between the claim date and the first check. The table $\mathcal{B}_{imn}$ is zero for all years n before benefits commence and for years after individual i 's plan horizon. It does not include spousal or survivor components, which are handled separately (Section 11.4).

11.2 Claiming-age adjustment factors

The benefit multiplier $\Phi(\alpha, \text{FRA})$ encodes the SSA rules for early reduction and delayed credits:

$$\Phi(\alpha, \text{FRA}) = \begin{cases} 1 + 0.08(\alpha - \text{FRA}), & \alpha \geq \text{FRA}, \\ 1 - \frac{1}{15}(\text{FRA} - \alpha), & \text{FRA} - 3 < \alpha < \text{FRA}, \\ 0.80 - \frac{1}{20}(\text{FRA} - 3 - \alpha), & \alpha \leq \text{FRA} - 3. \end{cases} \quad (11.3)$$

Delayed credits of 8% per year apply above FRA, capping at $f = 1.24$ at age 70 when FRA is 67. For early claiming the reduction is $\frac{1}{15}$ per year (i.e., 5/9 of 1% per month) for the first 3 years before FRA, and $\frac{1}{20}$ per year (i.e., 5/12 of 1% per month) for any additional years, yielding a minimum of approximately 70% of PIA at age 62. The Full Retirement Age FRA_i is determined from the individual's birth year as described in Chapter 2.

A separate factor applies to the spousal benefit, which receives no delayed credit beyond the spousal FRA (Section 11.4).

11.3 Binary claiming variables and LP constraints

The claiming-age decision for individual i is represented by K binary variables

$$z_{im}^{\text{ss}} \in \{0, 1\}, \quad i = 0, \dots, N_i - 1, \quad m = 0, \dots, K - 1, \quad (11.4)$$

where $z_{im}^{ss} = 1$ means individual i claims at age α_m . The *at-most-one* (AMO) constraint enforces that exactly one claiming age is selected per individual:

$$\sum_{m=0}^{K-1} z_{im}^{ss} = 1, \quad i = 0, \dots, N_i - 1. \quad (11.5)$$

The own Social Security benefit in year n for individual i is modeled as a continuous LP variable b_{in}^{ss} , linked to the binary selection through the precomputed table:

$$b_{in}^{ss} = \sum_{m=0}^{K-1} \mathcal{B}_{imn} z_{im}^{ss}, \quad i = 0, \dots, N_i - 1, \quad n = 0, \dots, N_n - 1. \quad (11.6)$$

Constraint (11.6) is an equality row added to the constraint matrix for each (i, n) pair, making b_{in}^{ss} a linear function of the binary claiming indicators.

Ineligible claiming ages (below the birth-date-adjusted minimum) are excluded by fixing the corresponding binary to zero in the variable bounds. For individuals who have already claimed before the optimization begins, the binary for the known claiming age is fixed to one and all others to zero, reducing the problem for that individual to a set of parameters with no additional free variables.

The total Social Security income entering the cash-flow constraint in year n is

$$\bar{\zeta}_{in} = b_{in}^{ss} + \delta_{in}^{sp}, \quad (11.7)$$

where b_{in}^{ss} is the LP variable for own benefits and δ_{in}^{sp} is a *spousal/survivor offset* parameter updated at each SC iteration (Section 11.4).

11.4 Spousal and survivor benefits in the SC loop

Spousal and survivor benefits depend on both partners' claiming ages in a non-linear fashion that is difficult to embed directly in an LP. Owl handles this through the self-consistent loop: the LP optimizes own benefits via z_{im}^{ss} , while spousal and survivor adjustments are carried as the fixed parameter δ_{in}^{sp} computed from the previous iteration's solution.

At the end of each SC iteration, the optimal claiming ages are extracted as

$$m_i^* = \arg \max_m z_{im}^{ss}, \quad i = 0, \dots, N_i - 1, \quad (11.8)$$

and the full benefit stream including spousal and survivor components is recomputed from those ages:

$$\bar{\zeta}_{in}^{\text{new}} = \text{compute_social_security_benefits}(\{m_i^*\}). \quad (11.9)$$

The spousal/survivor offset is then updated as

$$\delta_{in}^{\text{sp}} \leftarrow \bar{\zeta}_{in}^{\text{new}} - \mathcal{B}_{i, m_i^*, n}, \quad (11.10)$$

so that the difference between the total benefit and the own benefit at the chosen age is passed to the next LP iteration as a fixed income term. The variable $\bar{\zeta}_{in}$ in the plan is simultaneously updated to $\bar{\zeta}_{in}^{\text{new}}$ for use in taxability and MAGI computations.

Convergence and oscillation. The SC loop converges when the relative change in the objective falls below a tolerance. For single individuals the loop is typically monotonic; for couples the interdependence of spousal and survivor benefits can cause the solution to oscillate between two claiming-age combinations. An oscillation detector monitors the objective history: when a repeating cycle of length $c \geq 2$ is identified, the iteration with the best objective value within the cycle is selected as the final solution.

11.5 Implementation in Owl

The benefit table $\mathcal{B}_{imn}$ is assembled once per solve by `build\_own\_benefit\_table` in `socialsecurity.py` and stored as a pre-computed array of shape (N_i, K, N_n) . The binary variables z_{im}^{ss} are declared before the binary-block boundary in the variable map so that the AMO and benefit-definition constraints are included alongside the other bracket-selection binaries. Each SC iteration calls `compute\_social\_security\_benefits` to refresh the spousal/survivor offset. Partial optimization is supported: passing `withSSAges` a list containing a subset of individual names causes only those individuals' claiming ages to be optimized; others retain their user-specified values and contribute fixed benefit streams as in the base mode.

Limitations and assumptions

- Spousal and survivor benefits are not embedded directly in the LP. They are approximated through the SC loop, which may require several iterations for couple cases and can oscillate when the two partners' optimal ages are mutually dependent.
- The survivor benefit is assumed to be claimed in the year of the spouse's death; the strategy of deferring the survivor benefit to a later age is not optimized.
- The SSA family maximum benefit cap is not modeled; benefits may be overstated when multiple beneficiaries claim on one worker's record.

- The SSA earnings test is not modeled; all individuals are assumed to be fully retired before their FRA.
- The claiming age grid is monthly (97 points). Finer sub-monthly grids are not supported.

A. Source File Reference

The Owl Python package is located under `src/owlplanner/`. The table below summarizes the primary source files and their roles.

<code>plan.py</code>	The core optimization engine. Constructs and solves the LP/MIP using HiGHS (default, via <code>highspy</code>) or MOSEK directly. Implements the self-consistent iteration loop, all constraint-building methods, the objective functions, and post-solution analysis. Delegates stress-testing and stochastic spending methods to <code>stresstests.py</code> via mixin. This is the largest and most complex file.
<code>varmap.py</code>	Variable block indexing and extraction for the LP solver. <code>VarBlock</code> records the start offset and shape of one decision-variable family (e.g., $b, d, e, f, g, h, m, s, w, x, z^m, \dots$); <code>VarMap</code> accumulates them in declaration order and exposes a cursor tracking the running offset. Used by <code>plan.py</code> to build and query all LP/MIP variable indices without hard-coded offset arithmetic.
<code>abcapi.py</code>	A solver-neutral constraint-building API used by <code>plan.py</code> . Provides an abstract interface for adding rows to the constraint matrix, keeping the model description independent of any specific LP solver.
<code>stresstests.py</code>	Stress-testing and stochastic spending optimization methods. Provides <code>run_historical_range</code> , <code>run_mc</code> , and <code>run_stochastic_spending</code> , which take a <code>Plan</code> instance and run the optimizer across many return scenarios using a thread pool. Also implements the efficient-frontier LP, CVaR computation, and the Retirement Efficiency Score (RES). <code>Plan</code> exposes these as methods by delegation (mixin pattern).

`tax_federal.py`

Federal tax calculations: ordinary income brackets and standard deductions, LTCG tax thresholds, SS taxability formula, NIIT, ACA Premium Tax Credit (with `acaCosts` and `acaVals`), and Medicare IR-MAA premiums and bracket thresholds. Also computes RMD fractions from the IRS Uniform Lifetime Table. Implements the OBBBA (2025) tax schedule, the 65+ bonus deduction phaseout, and the future-law reversion scenario (`y0BBBA` parameter).

`tax_state.py`

State income tax parameters for all 50 states and the District of Columbia. Loads per-state, per-filing-status bracket structures, standard deductions, Social Security treatment, and retirement-income and pension exemption caps from `data/taxes_state.toml`, and converts them into the inflation-indexed LP parameter arrays used by the state-tax constraints (Sec. (5.1.17)).

`socialsecurity.py`

Social Security benefit modeling: FRA schedules, own-benefit and spousal adjustment factors, spousal and survivor benefit computation, and optional benefit trim for policy scenarios. When `withSSAges="optimize"` is selected, `build\_own\_benefit\_table` precomputes the $N_i \times 97 \times N_n$ benefit table covering all monthly claiming ages from 62 to 70 (97 choices), used by the MIP claiming-age optimizer.

`timelists.py`

Reads and validates the Wages and Contributions time-horizon spreadsheet (one tab per individual) and the Fixed Assets and Debts tables. Returns year-indexed data structures consumed by `plan.py`.

`fixedassets.py`

Models fixed-income assets (real estate, annuities, collectibles, etc.), computing year-by-year income, proceeds, and capital gains arising from assets listed in the Fixed Assets table.

`debts.py`

Models debt obligations (loans, mortgages), computing annual payment schedules from the Debts table.

`rates.py`

Loads and pre-processes the historical market-return data from `data/rates.csv` (equity, bond, and inflation series from 1928 to present) and provides utilities for constructing rate scenarios.

<code>rate_models/</code>	A pluggable system of rate model classes inheriting from <code>BaseRateModel</code> . Built-in models (<code>historical</code> , <code>historical_average</code> , <code>historical_gaussian</code> , <code>historical_lognormal</code> , <code>gaussian</code> , <code>lognormal</code> , <code>historical_bootstrap</code> , <code>vector_ar</code> , <code>garch_dcc</code> , and <code>user</code>) are in <code>builtin.py</code> and <code>_builtin_impl.py</code> . Custom models are discoverable at runtime if they subclass <code>BaseRateModel</code> and are importable.
<code>config/schema.py</code>	Pydantic v2 models defining the full case configuration schema. All plan parameters (account balances, claiming ages, rate method, options, etc.) are validated here before being passed to <code>plan.py</code> .
<code>config/toml_io.py</code>	TOML file loading and saving for <code>.toml</code> case files. Also handles backward compatibility for older file formats.
<code>plotting/</code>	Visualization backends: <code>plotly_backend.py</code> for interactive plots in the Streamlit UI, and <code>matplotlib_backend.py</code> for static plots used in the CLI and Jupyter notebooks.
<code>cli/</code>	A Click-based command-line interface registered as the <code>owlcli</code> entry point. Supports <code>list</code> (enumerate available case files) and <code>run</code> (solve a case from a TOML file) commands.
<code>assistant/</code>	AI-assistant integration layer shared by the MCP server and the chat page. <code>tools.py</code> implements the MCP tool functions (case solving, comparison, stochastic frontiers, <code>explain_results</code> , ...); <code>explain.py</code> classifies the constraint tags (Sec. (7.4)) and turns a solved plan's shadow prices and primal solution into a structured, JSON-ready explanation; <code>intake.py</code> provides the guided intake prompt and resources.

References

- [1] M.-D. Lacasse, “Optimal Wealth Lab (Owl),” GitHub repository, 2025. <https://github.com/mdlacasse/Owl>.
- [2] Social Security Administration, *Program Operations Manual System (POMS)*, RS 00615.003 (Full Retirement Age), RS 00615.015 (SSA age computation), <https://secure.ssa.gov/poms.nsf/lnx/0300615003>.
- [3] Internal Revenue Service, *Publication 915: Social Security and Equivalent Railroad Retirement Benefits*, <https://www.irs.gov/pub/irs-pdf/p915.pdf>.
- [4] Internal Revenue Code § 408A, “Roth Individual Retirement Accounts,” <https://www.law.cornell.edu/uscode/text/26/408A>.
- [5] Internal Revenue Service, *Publication 590-B: Distributions from Individual Retirement Arrangements (IRAs)*, <https://www.irs.gov/pub/irs-pdf/p590b.pdf>.
- [6] Internal Revenue Code § 223, “Health Savings Accounts,” <https://www.law.cornell.edu/uscode/text/26/223>.
- [7] Internal Revenue Service, *Publication 969: Health Savings Accounts and Other Tax-Favored Health Plans*, <https://www.irs.gov/pub/irs-pdf/p969.pdf>.
- [8] Internal Revenue Service, *Form 8889: Health Savings Accounts (HSAs)*, <https://www.irs.gov/pub/irs-pdf/f8889.pdf>.
- [9] Internal Revenue Code § 36B, “Refundable credit for coverage under a qualified health plan,” <https://www.law.cornell.edu/uscode/text/26/36B>.
- [10] Internal Revenue Service, *Rev. Proc. 2024-35*, “Applicable Percentage Table for the Premium Tax Credit for Taxable Years Beginning in Calendar Year 2025,” 89 Fed. Reg. 66278 (2024).

- [11] Internal Revenue Service, *Rev. Proc. 2025-25*, “Applicable Percentage Table for the Premium Tax Credit for Taxable Years Beginning in Calendar Year 2026,” <https://www.irs.gov/pub/irs-drop/rp-25-25.pdf>.
- [12] U.S. Department of Health and Human Services, “Federal Poverty Guidelines” (annual), <https://aspe.hhs.gov/poverty-guidelines>.
- [13] Centers for Medicare & Medicaid Services, “2026 Medicare Parts A & B Premiums and Deductibles,” <https://www.cms.gov/newsroom/fact-sheets/2026-medicare-parts-b-premiums-deductibles>.
- [14] Internal Revenue Service, “Topic No. 409: Capital Gains and Losses,” <https://www.irs.gov/taxtopics/tc409>.
- [15] Internal Revenue Code § 1411, “Imposition of Tax (Net Investment Income),” <https://www.law.cornell.edu/uscode/text/26/1411>.
- [16] Internal Revenue Service, *Publication 17: Your Federal Income Tax*, <https://www.irs.gov/pub/irs-pdf/p17.pdf>.
- [17] One Big Beautiful Bill Act (OBBBA), signed July 4, 2025. Extends and modifies individual income tax provisions; adds § 1002 senior deduction.
- [18] SECURE 2.0 Act of 2022, Pub. L. No. 117-328, Division T, 136 Stat. 5966.
- [19] R. J. Shiller, *Irrational Exuberance*, Princeton University Press, Princeton, NJ, 2000. Historical return data available at <http://www.econ.yale.edu/~shiller/data.htm>.
- [20] J. C. Hull, *Options, Futures, and Other Derivatives*, 10th ed., Pearson, Hoboken, NJ, 2017. (Chapters 15–16 derive the log-normal / GBM return model and the arithmetic-to-log-space parameter conversion.)
- [21] B. Efron, “Bootstrap methods: another look at the jackknife,” *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979.
- [22] H. R. Künsch, “The jackknife and the bootstrap for general stationary observations,” *The Annals of Statistics*, vol. 17, no. 3, pp. 1217–1241, 1989.
- [23] D. N. Politis and J. P. Romano, “A circular block resampling procedure for stationary data,” in *Exploring the Limits of Bootstrap*, R. LePage and L. Billard, Eds. Wiley, New York, 1992, pp. 263–270.

- [24] D. N. Politis and J. P. Romano, "The stationary bootstrap," *Journal of the American Statistical Association*, vol. 89, no. 428, pp. 1303–1313, 1994.
- [25] J. Y. Campbell and L. M. Viceira, *Strategic Asset Allocation: Portfolio Choice for Long-Term Investors*, Oxford University Press, Oxford, 2002.
- [26] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*, Springer-Verlag, Berlin, 2005.
- [27] R. F. Engle, "Dynamic conditional correlation: a simple class of multivariate generalized autoregressive conditional heteroskedasticity models," *Journal of Business & Economic Statistics*, vol. 20, no. 3, pp. 339–350, 2002.
- [28] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.
- [29] G. J. McLachlan and D. Peel, *Finite Mixture Models*. Wiley, New York, 2000.
- [30] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.
- [31] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [32] A. Sklar, "Fonctions de répartition à n dimensions et leurs marges," *Publications de l'Institut de Statistique de l'Université de Paris*, vol. 8, pp. 229–231, 1959.
- [33] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.
- [34] H. M. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.
- [35] W. P. Bengen, "Determining withdrawal rates using historical data," *Journal of Financial Planning*, vol. 7, no. 4, pp. 171–180, 1994.
- [36] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical Finance*, vol. 9, no. 3, pp. 203–228, 1999.

- [37] R. T. Rockafellar and S. Uryasev, “Optimization of conditional value-at-risk,” *Journal of Risk*, vol. 2, no. 3, pp. 21–41, 2000.
- [38] J. F. Benders, “Partitioning procedures for solving mixed-variables programming problems,” *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.