Worked examples

Contents

mortgagemath 0.6.0 · rendered 2026-05-02

This vignette catalogs the loan types `mortgagemath` reproduces to the cent against published sources, organized by country convention. Each example shows: a one-paragraph **scenario**, the exact `LoanParams(...)` configuration, the equivalent `mortgagemath` CLI invocation, a live Python chunk emitting the schedule and the published anchors it matches, and a citation linking to both the source document and the fixture file.

For the broader institutional and mathematical history see the *History* vignette; for the validated worked-example matrix see *Validation*; for a 60-second orientation see *At a glance*.

United States

30-year fixed-rate residential (CFPB H-25(B))

Scenario. The Consumer Financial Protection Bureau's sample TILA-RESPA Integrated Disclosure form H-25(B), a fictional "Ficus Bank" closing-disclosure example. The published monthly P&I of $761.78 implies ROUND_HALF_UP rounding; the unrounded closed-form value is $761.7840…, which would round up to $761.79 under the library's default ROUND_UP.

```
mortgagemath payment --principal 162000 --rate 3.875 --term-months 360 \
    --payment-rounding ROUND_HALF_UP --interest-rounding ROUND_HALF_UP
```

```
from decimal import Decimal
from mortgagemath import LoanParams, PaymentRounding, periodic_payment

cfpb = LoanParams(
    principal=Decimal("162000.00"),
    annual_rate=Decimal("3.875"),
```

```
    term_months=360,
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
)
print(f"Monthly P&I: ${periodic_payment(cfpb)}  (CFPB H-25(B): $761.78)")
```

```
Monthly P&I: $761.78  (CFPB H-25(B): $761.78)
```

Source. CFPB Closing Disclosure Sample H-25(B) (2014). Fixture: `cfpb_h25b_ficus_30yr_3875_162000.{toml,csv}`.

15-year fixed-rate residential (OpenStax, ROUND_UP)

Scenario. OpenStax *Contemporary Mathematics* §6.8 home-loan example. The textbook explicitly states *"payment to lenders is always rounded up to the next penny"* — confirming the library's default ROUND_UP convention used by most US residential lenders.

```
openstax = LoanParams(
    principal=Decimal("136700.00"),
    annual_rate=Decimal("5.75"),
    term_months=180,
    # ROUND_UP is the default; shown explicitly for clarity.
    payment_rounding=PaymentRounding.ROUND_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
)
print(f"Monthly P&I: ${periodic_payment(openstax)}  (OpenStax: $1,135.18)")
```

```
Monthly P&I: $1135.18  (OpenStax: $1,135.18)
```

Source. OpenStax *Contemporary Mathematics* §6.8, CC-BY-4.0. Fixture: `openstax_contemp_home_180mo_575_136700.{toml,csv}`.

Commercial Actual/360 with balloon (Fannie Mae §1103)

Scenario. Fannie Mae *Multifamily Selling and Servicing Guide* §1103, the Tier 2 SARM worked example. $25 M / 5.5% / 10-year term on a 30-year amortization basis with an Actual/360 day-count convention. The borrower makes 120 monthly payments at the closed-form level, then owes a published balloon of $20,885,505.83 at term.

```
mortgagemath schedule --principal 25000000 --rate 5.5 --term-months 120 \
    --amortization-period-months 360 --day-count actual/360 \
    --start-date 2018-12-01 \
    --payment-rounding ROUND_HALF_UP --interest-rounding ROUND_HALF_UP \
    --format csv > fnma_1103.csv
```

```
from datetime import date
from mortgagemath import DayCount, amortization_schedule

fnma = LoanParams(
    principal=Decimal("25000000.00"),
    annual_rate=Decimal("5.5"),
    term_months=120,
    amortization_period_months=360,
    day_count=DayCount.ACTUAL_360,
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
    start_date=date(2018, 12, 1),
)
sched = amortization_schedule(fnma)
```

```
print(f"Monthly P&I:                    ${periodic_payment(fnma):,}")
print(f"Balloon at term-120:            ${sched[120].balance:,}")
print(f"  (Fannie Mae §1103 published: $141,947.25 / $20,885,505.83)")
```

```
Monthly P&I:                 $141,947.25
Balloon at term-120:         $20,885,505.83
  (Fannie Mae §1103 published: $141,947.25 / $20,885,505.83)
```

Source. Fannie Mae Multifamily Guide §1103 (eff. 2026-04-03). Fixture: `fanniemae_mf_1103_25m_550_360mo.{toml,csv}`.

Adjustable-rate mortgage (Reg Z H-14)

Scenario. 12 CFR Part 1026 Appendix H Sample H-14 — the Federal-Reserve-promulgated regulatory ARM disclosure. $10,000 / 30-year term / fully-amortizing 1/1 ARM at 1-year CMT + 3 pp margin, 2 pp annual periodic cap, 5 pp symmetric lifetime cap. The published example traces 15 years of historical adjustments (1982–1996) using actual 1-year CMT values, with the lifetime floor at 12.41% binding from 1986 onward.

Yr	Year	CMT index	Fully indexed	Cap binds?	Effective rate
1	1982	14.41%	17.41%	(initial)	17.41%
2	1983	9.78%	12.78%	periodic (down)	15.41%
3	1984	12.17%	15.17%	—	15.17%
4	1985	7.66%	10.66%	periodic (down)	13.17%
5	1986	6.36%	9.36%	lifetime (floor)	12.41%
6+	1987–1996	varies	≤ 9.36%	lifetime holds	12.41%

```
mortgagemath schedule --principal 10000 --rate 17.41 --term-months 360 \
    --payment-rounding ROUND_HALF_UP --interest-rounding ROUND_HALF_UP \
    --rate-change 13:15.41 --rate-change 25:15.17 --rate-change 37:13.17 \
    --rate-change 49:12.41  # ... and similarly for years 6-15
```

```
from mortgagemath import RateChange, BalanceTracking

rates = ("15.41", "15.17", "13.17", "12.41", *(["12.41"] * 10))

regz_h14 = LoanParams(
    principal=Decimal("10000"),
    annual_rate=Decimal("17.41"),
    term_months=360,
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
    balance_tracking=BalanceTracking.ROUND_EACH,
    rate_schedule=tuple(
        RateChange(
            effective_payment_number=12 * (yr - 1) + 1,
            new_annual_rate=Decimal(rate),
        )
        for yr, rate in enumerate(rates, start=2)
    ),
)
sched = amortization_schedule(regz_h14)
```

Anchor	Library	Published	Match
first row, year 1 (pmt 1)	$145.90	$145.90	✓
balance after year 1 (pmt 12)	$9,989.37	$9989.37	✓
first row, year 2 (pmt 13)	$129.81	$129.81	✓
first row, year 5 (pmt 49)	$106.73	$106.73	✓
balance after year 5 (pmt 60)	$9,848.94	$9848.94	✓
balance after year 15 (pmt 180)	$8,700.37	$8700.37	✓

Source. 12 CFR Part 1026 Appendix H Sample H-14. Fixture: `regz_apph_h14_arm_10k_1741_360mo.{toml,csv}`.

ARM with payment cap + negative amortization (ProEducate)

Scenario. A *payment cap* is distinct from a *rate cap*. A rate cap bounds the interest rate; a payment cap bounds the dollar payment directly, regardless of what the new rate would otherwise produce. When the cap binds *and* the new periodic interest exceeds the capped payment, the unpaid interest is capitalized into the balance — the loan goes into negative amortization. ProEducate publishes a worked $65,000 / 10% rising to 12% / 30-year / 7.5% annual cap example; the library reproduces every cell including the explicit $420.90 of year-2 cumulative negative amortization.

```
mortgagemath schedule --principal 65000 --rate 10 --term-months 360 \
    --payment-rounding ROUND_HALF_UP --interest-rounding ROUND_HALF_UP \
    --rate-change 13:12:cap=1.075 --format csv
```

```
proeducate = LoanParams(
    principal=Decimal("65000"),
    annual_rate=Decimal("10"),
    term_months=360,
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
    balance_tracking=BalanceTracking.ROUND_EACH,
    rate_schedule=(
        RateChange(
            effective_payment_number=13,
            new_annual_rate=Decimal("12"),
            payment_cap_factor=Decimal("1.075"),  # 7.5% annual cap
        ),
    ),
)
sched = amortization_schedule(proeducate)
```

- Year 1 P&I (closed-form at 10%): **$570.42** (published $570.42)
- Balance after pmt 12: **$64,638.72** (published $64,638.72)
- Year 2 capped P&I (= year-1 × 1.075): **$613.20** (published $613.20)
- Year 2 monthly principal (negative): **$−33.19** (published −$33.19)
- Cumulative neg-am over year 2: **$420.90** (published $420.90)
- Balance after pmt 24: **$65,059.62** (published $65,059.62)

Source. ProEducate *ARM Payment Caps*, 2014. Fixture: `proeducate_arm_pmt_cap_65k_10pct_to_12pct_360mo.{toml,csv}`.

Carry-precision graduate-CRE (Geltner Ch 20)

Scenario. Geltner et al., *Commercial Real Estate Analysis and Investments*, online supplement Chapter 20 Exhibit 20-6 — a $1 M / 12% / 30-year constant-payment mortgage. The textbook uses Excel-default carry-precision balance tracking throughout, so the library's `CARRY_PRECISION` mode is the right configuration.

```
geltner = LoanParams(
    principal=Decimal("1000000.00"),
    annual_rate=Decimal("12"),
    term_months=360,
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
    balance_tracking=BalanceTracking.CARRY_PRECISION,
)
print(f"Monthly P&I: ${periodic_payment(geltner):,}  (Geltner: $10,286.13)")
```

```
Monthly P&I: $10,286.13  (Geltner: $10,286.13)
```

Source. Geltner, Miller, van de Minne, Eichholtz, Lindenthal, & Shen, *Commercial Real Estate Analysis and Investments*, Routledge, 2024. Fixture: `geltner_ch20_cpm_1m_1200_360mo.{toml,csv}`.

Half-cent rounding boundary (synthetic)

Scenario. Three paired synthetic fixtures engineered so month-1 unrounded interest equals exactly $400.005 — a half-cent boundary that distinguishes ROUND_HALF_UP from ROUND_HALF_EVEN and the closed-form payment from ROUND_UP. $100,001.25 / 4.80% / 30-year. Same loan; three rounding configurations.

```
synth_principal = Decimal("100001.25")
synth_rate = Decimal("4.80")

for rounding, expected in [
    (PaymentRounding.ROUND_HALF_UP, "524.67"),
    (PaymentRounding.ROUND_UP,      "524.68"),
]:
    loan = LoanParams(
        principal=synth_principal,
        annual_rate=synth_rate,
        term_months=360,
        payment_rounding=rounding,
        interest_rounding=PaymentRounding.ROUND_HALF_UP,
    )
    print(f"{rounding.value:14s} → ${periodic_payment(loan)}  "
          f"(expected ${expected})")
```

```
ROUND_HALF_UP  → $524.67  (expected $524.67)
ROUND_UP       → $524.68  (expected $524.68)
```

Sources. Synthetic boundary fixtures. `synthetic_halfcent_halfup_360_480_100001p25.{toml,csv}` and companions.

Effective-annual on monthly cadence (Skinner 1913 piano)

Scenario. Skinner's 1913 *Mathematical Theory of Investment* §42 Example 3 — a piano costing $500 to be paid off at 6% *effective annual* over 5 years monthly. The actuarial convention treats 6% as the effective-annual rate and derives the equivalent nominal-monthly rate $i^{(12)} = 12 \cdot \left((1.06)^{1/12} - 1\right) \approx 0.0584$, *not* 6/12 = 0.5%. Naïve `Compounding.MONTHLY` gives $9.67 — a 4-cent miss. The library's `Compounding.ANNUAL` mode reproduces Skinner's published $9.63 exactly.

```
from mortgagemath import Compounding, PaymentFrequency

skinner_piano = LoanParams(
    principal=Decimal("500.00"),
    annual_rate=Decimal("6"),
    term_months=60,
    payment_frequency=PaymentFrequency.MONTHLY,
    compounding=Compounding.ANNUAL,
```

```
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
)
print(f"Monthly piano payment: ${periodic_payment(skinner_piano)}  "
      f"(Skinner §42 Ex 3: $9.63)")
```

```
Monthly piano payment: $9.63  (Skinner §42 Ex 3: $9.63)
```

Source. Skinner, *The Mathematical Theory of Investment*, Boston: Ginn and Company, 1913. Public domain on Internet Archive. Fixture: `skinner_1913_42_ex3_500_6pct_5yr_monthly.{toml,csv}`.

Annual cadence + annual compounding (Arcones SOA FM)

Scenario. Miguel Arcones's *Manual for SOA Exam FM / CAS Exam 2*, §4.1 Example 4 — $20,000 / 8% effective annual / 12 annual payments. The full 12-row schedule is published in the source; the library reproduces every cell exactly under `Compounding.ANNUAL` + `PaymentFrequency.ANNUAL`.

```
arcones = LoanParams(
    principal=Decimal("20000.00"),
    annual_rate=Decimal("8"),
    term_months=144,
    payment_frequency=PaymentFrequency.ANNUAL,
    compounding=Compounding.ANNUAL,
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
    balance_tracking=BalanceTracking.ROUND_EACH,
)
sched = amortization_schedule(arcones)
print(f"Annual installment:       ${periodic_payment(arcones):,.2f}")
print(f"Final-row trueup (yr 12):  ${sched[12].payment}  (Arcones: $2,653.91)")
```

```
Annual installment:       $2,653.90
Final-row trueup (yr 12):  $2653.91  (Arcones: $2,653.91)
```

Source. Arcones, *Manual for SOA Exam FM / CAS Exam 2* §4.1, 2009. Fixture: `arcones_soa_fm_4_1_ex4_20k_8pct_12yr_annual.{toml,csv}`.

Given-payment, find-term (FHLBB 1935 Plan A) — *new in v0.6.0*

Scenario. The Federal Home Loan Bank Board's *FHLBB Review* of March 1935 published the earliest U.S. federal-authority worked direct-reduction amortization schedule, in an article recommending the direct-reduction plan over the dominant share-accumulation B&L scheme. Plan A: $3,000 at 6% with *monthly* interest credit, payment chosen by convention as 1% of original principal = $30.00 per month. The schedule retires in 138 full payments of $30 plus a 139th payment of $29.27.

This is the historical "given-payment, find-term" convention: the lender chose a round payment (typically a percentage of original principal) and accepted whatever final-payment trueup the math produced. The library's default closed-form mode (term in, payment out) cannot reproduce this — it derives a level $29.9964 for term=139, which rounds to $30.00 with a $29.35 final-row trueup. The published $29.27 is the carry-precision balance after 138 payments × (1 + r), rounded once.

v0.6.0's `payment_override` field pins the payment and applies a "round-the-total" final-row trueup, reproducing every published cell.

```
mortgagemath schedule --principal 3000 --rate 6 --term-months 139 \
    --payment-rounding ROUND_HALF_UP --interest-rounding ROUND_HALF_UP \
    --balance-tracking carry_precision \
    --payment-override 30 --format table
```

```
fhlbb = LoanParams(
    principal=Decimal("3000.00"),
    annual_rate=Decimal("6"),
    term_months=139,
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
    balance_tracking=BalanceTracking.CARRY_PRECISION,
    payment_override=Decimal("30.00"),
)
sched = amortization_schedule(fhlbb)
for n in (1, 138, 139):
    inst = sched[n]
    print(f"month {n:>3}: payment=${inst.payment} "
          f"interest=${inst.interest} principal=${inst.principal} "
          f"balance=${inst.balance}")
print(f"\nFinal payment: ${sched[139].payment}  (FHLBB published $29.27)")
```

```
month   1: payment=$30.00 interest=$15.00 principal=$15.00 balance=$2985.00
month 138: payment=$30.00 interest=$0.29 principal=$29.71 balance=$29.13
month 139: payment=$29.27 interest=$0.15 principal=$29.12 balance=$0.00

Final payment: $29.27  (FHLBB published $29.27)
```

Source. FHLBB *Federal Home Loan Bank Review*, Vol. 1 No. 6, March 1935, pp. 187–198, public domain. Fixture: `fhlbb_1935_plan_a_3k_6pct_30_per_mo.{toml,csv}`.

Canada

Semi-annual *j_2* with monthly payments (Olivier Chans)

Scenario. Canadian residential mortgages quote rates as *semi-annually compounded* (j_2), per Section 6 of the federal *Interest Act*. A US "5% / 30 yr" loan and a Canadian "5% / 30 yr" loan have different periodic rates and different monthly payments even though both quote 5%. Olivier's *Business Math* §13.4 walks through the Chans family's first term: $350,100 / j_2 = 4.9% / 3-year fixed term on a 20-year amortization basis, monthly payments. At end of the 3-year term the unpaid balance is the balloon to be renewed at then-prevailing rates.

> **i Note**
>
> **US**: monthly periodic rate is `r/12` directly. A 5% loan uses $5/1200 = 0.004167$ per month.
>
> **Canadian (j_2)**: the equivalent monthly periodic rate is $(1 + j_2/200)^{2/12} - 1$. For `j_2 = 5%`, this is $(1.025)^{1/6} - 1 \approx 0.41239\%$, *not* $5/12 \approx 0.4167\%$.

```
mortgagemath payment --principal 350100 --rate 4.9 --term-months 36 \
    --amortization-period-months 240 --compounding semi_annual \
    --payment-rounding ROUND_HALF_UP --interest-rounding ROUND_HALF_UP
```

```
chans = LoanParams(
    principal=Decimal("350100"),
    annual_rate=Decimal("4.9"),
    term_months=36,
    amortization_period_months=240,
    compounding=Compounding.SEMI_ANNUAL,
    payment_frequency=PaymentFrequency.MONTHLY,
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
```

```
)
print(f"Monthly P&I:                           ${periodic_payment(chans):,}")
sched = amortization_schedule(chans)
print(f"Balance after pmt 36 (renewal balloon): ${sched[36].balance:,}")
print(f"  (Olivier published: $2,281.73 / $316,593.49)")
```

```
Monthly P&I:                           $2,281.73
Balance after pmt 36 (renewal balloon): $316,593.49
  (Olivier published: $2,281.73 / $316,593.49)
```

Source. Olivier, *Business Math: A Step-by-Step Handbook* §13.4, LibreTexts (CC-BY-NC-SA), 2021. Fixture: `olivier_chans_350100_490_36mo_term_240mo.{toml,csv}` plus the companion `olivier_chans_renewal_316593p49_585_204mo.{toml,csv}` for the renewal-term scenario at 5.85%.

Semi-annual *j_2* with quarterly payments (eCampus §4.4.1)

Scenario. Canadian semi-annual compounding combines naturally with non-monthly payment cadences. eCampus Ontario's *Mathematics of Finance* §4.4.1 publishes a **quarterly-payment** worked example: $297,500 / j_2 = 3.8% / 3-year fixed term on a 20-year amortization, quarterly payments.

```
ecampus_q = LoanParams(
    principal=Decimal("297500"),
    annual_rate=Decimal("3.8"),
    term_months=36,
    amortization_period_months=240,
    compounding=Compounding.SEMI_ANNUAL,
    payment_frequency=PaymentFrequency.QUARTERLY,
    payment_rounding=PaymentRounding.ROUND_HALF_UP,
    interest_rounding=PaymentRounding.ROUND_HALF_UP,
)
sched = amortization_schedule(ecampus_q)
print(f"Quarterly P&I:              ${periodic_payment(ecampus_q):,}")
print(f"Balance after 12 quarters:   ${sched[12].balance:,}")
print(f"  (eCampus published: $5,317.62 / $265,830.61)")
```

```
Quarterly P&I:              $5,317.62
Balance after 12 quarters:   $265,830.61
  (eCampus published: $5,317.62 / $265,830.61)
```

The schedule walks 12 quarterly rows (not 12 monthly), because $\text{term\_months} \cdot \text{payments\_per\_year} / 12 = 36 \cdot 4/12 = 12$ payments.

Source. eCampus Ontario *Mathematics of Finance* §4.4.1, CC-BY-4.0. Fixture: `ecampus_finmath_4_4_1_297500_380_q_36mo_term_240mo.{toml,csv}` plus the renewal companion at `j_2` = 2.5%.

Monthly compounding on Canadian textbook examples

The same eCampus textbook also publishes US-convention examples (monthly compounding, monthly payments) — these appear in §4.3 Examples 4.3.1 (Pearline, $10,000 / 10% effective annual / 4-year annual, full schedule), Exercise 2 (Erika, $32,600 / 4.83% / 9-year monthly with year-aggregate anchors), and Exercise 3 (Johnetta, $20,200 / 3.53% / 8-year monthly with a mid-schedule probe at payment 60). All three reproduce to the cent under the library's defaults; see the *Validation* vignette for the per-fixture matrix.

ℹ Note

Why this matters. Without `Compounding.SEMI_ANNUAL`, a Canadian-style mortgage calculator written in stdlib Python will quote a payment about 0.7% high on a typical 25-year loan — a few dollars per month, but several thousand dollars over the loan life. Most Python amortization libraries silently get this wrong.

France

The 1852 Crédit Foncier de France structure embeds an administrative loading (frais d'administration + fonds de réserve + impôt = 0.50 fr. per 100 fr. principal per year on a 5.50-fr. interest+amortization base) into the published *annuité* of 6.00 fr. per 100 fr. — a structure the library's pure interest+principal model cannot, at present, reproduce. Modern French *tableaux d'amortissement* by contrast separate *capital amorti*, *intérêts*, and *assurance emprunteur* into distinct columns; the *capital + intérêts* slice is matchable under the library's existing closed-form at standard nominal rates with no extension.

A planned `fee_per_period` field on `LoanParams` (see *History* §10) would let the library reproduce CF-style combined *annuités* and any modern French published table that shows only a *total échéance* with insurance loaded in. Shipping the feature is conditional on first retrieving one verifiable row-level CF or modern-French source the library can match cell-for-cell. No France fixture currently ships in `mortgagemath`.

United Kingdom

The British **Benefit Building Societies Act 1836** provided the statutory frame for what would become the U.K.'s dominant retail mortgage lender through most of the twentieth century. From the 1990s onward, U.K. building-society direct-reduction mortgages converge with the U.S. convention; a U.K. fixture is mechanically a duplicate of an existing U.S. example. This section reserves the slot until a specific U.K. published table surfaces.

Australia

Victoria's **Credit Foncier Act 1896** established a state-owned mortgage bank on the CF model, and the term *credit foncier* persisted in Australian mortgage-banking parlance for decades thereafter. A 1900s–1950s state-bank schedule with fee loading would land here, contingent on the planned `fee_per_period` extension shipping and a verifiable source.