令和 7 年度厚生労働行政推進調査事業費補助金(政策科学総合研究事業(政策科学推進研究事業))

社会保障給付に関するマイクロシミュレーション分析の研究 分担研究報告書

公的年金財政検証プログラムの Python への移植

川出真清[1]・佐藤格[2]・八塩裕之[3]・金田陸幸[4]

2026 年 5 月 31 日

第 1 節：はじめに

1.1 背景

公的年金の財政検証は、人口・経済前提のもとで将来の収支見通しを試算し、制度の持続可能性や調整の必要性を検討するための基盤である。厚生労働省が公表する財政検証の考え方や結果の整理は、同省の資料 (厚生労働省, 2024a, 2024b) に示されている。

一方、試算を担う実行プログラムは長年の改修を経て、C / C++ の実行ファイルが連鎖し、中間結果を多数の CSV で受け渡す構成になっている。制度改正のたびに該当箇所の解読と再コンパイルが必要になり、処理の全体像が分かりにくいことが、運用・研究の双方で障壁になってきた。

本プロジェクト（rev2024）は、この財政検証の計算ロジックを Python で再実装し、可読性・検証可能性・将来の分析基盤（データサイエンス環境、マイクロシミュレーション連携など）との接続を目指している。

なお、本稿は、次の二つの読者を想定して書いている。

[1] 日本大学経済学部教授

[2] 国立社会保障・人口問題研究所社会保障基礎理論研究部第 1 室長

[3] 京都産業大学経済学部教授

[4] 神戸学院大学経済学部准教授

- 財政検証の進捗だけを把握したい方: 財政検証移植プロジェクトが「何をしているか」「どこまで終わっているか」を、専門用語を抑えて把握できること。
- 年金財政に詳しく、プログラム実装にはあまり触れない方: 公式の財政検証の計算の流れ（モジュールの役割）を、政策・数理の観点から理解できること。

数式やソースコードの細部は、リポジトリ内の docs/PARITY_OVERVIEW_rev2024.md・docs/PARITY_DEBUG_GUIDE.md・docs/CURRENT_ISSUES.md に委ねる。本稿は状況の要約と構造の説明に重点を置く。

1.2 本作業の目的

本プロジェクトは、主に三つの問いを目的として構成されている。第一に「なぜ Python による再実装が必要か」という点である。これは、旧来の C / C++ からなるレガシー実行系の保守負荷を下げ、単一言語で一連の試算を透過的に扱えるようにするためである。第二に「計算結果は信頼できるか」という検証の問いである。これに対しては、高橋他(2023)による R をラッパーとして介して実行した従来の公式モデルの出力（ゴールデンデータ）と直接突き合わせる手法を採り、実装漏れを特定・解消したうえで、実務上許容できる水準に達しているかを確認する。第三に「将来何ができるようになるか」という展望の問いである。Python のエコシステムを活用することで、パラメータ変更を伴う反復試算や高度な可視化、さらには将来的なミクロデータとの接続など、分析の拡張余地を確保することが狙いである。

1.3 構成

第 2 節で財政検証プログラムの構造（年金専門家向け）と、公式 C・R・Python の対照表（モデル作成者・プログラム作成者向け）、第 3 節で参照系の位置づけ、第 4 節で検証の考え方と現在の進捗（初見向け）、第 5 節で精度・長期収支、第 6 節で結論と展望を述べる。

第 2 節：財政検証プログラムの構造（年金財政の観点）

財政検証とは、国民年金法第 4 条第 3 項および厚生年金保険法第 2 条第 4 項に基づき、少なくとも 5 年ごとに実施が義務付けられた、公的年金制度の財政の現況および将来の見通し

の作成である（厚生労働省, 2024a, 2024b）。2004（平成 16）年改正により、保険料固定方式と自動調整メカニズム（マクロ経済スライド）が導入されたことに伴い、財政検証は制度の長期的均衡を確認する定期検証として制度化された。直近の 2024（令和 6）年財政検証は、厚生労働省年金局数理課が実施・公表し、複数の経済シナリオのもとで 2060 年代までの所得代替率や積立金の推移を試算している（厚生労働省, 2024a）。

財政検証の問いは本質的に 2 つである。

1. 給付水準の持続可能性: 現行制度を前提とした場合、将来の所得代替率は 50%を下回るか。
2. 財政均衡期間: マクロ経済スライドによる給付調整はいつ終了するか（調整終了 = 財政均衡）。

この 2 つの問いに答えるために、約 100 年先（2120 年頃）までの収支見通しが試算される。

公的年金財政の三勘定構造

財政検証を理解するうえで、公的年金財政が三つの勘定から構成されることを把握しておく必要がある（厚生労働省, 2024a）。国の特別会計として、厚生年金勘定・国民年金勘定・基礎年金勘定の三つが設けられており、それぞれの収支は以下のように構成される。

厚生年金勘定は保険料収入・運用収入・国庫負担（1/2 相当）を収入とし、基礎年金拠出金と報酬比例部分の厚生年金給付を主な支出とする。国民年金勘定は同様の構成に加え、付加年金等の独自給付を支出とする。基礎年金勘定は、国民年金・厚生年金の両勘定からの基礎年金拠出金のみを収入とし、基礎年金給付費を支出とする完全賦課方式の勘定で積立金を持たない。

計算上の連結点: 基礎年金拠出金は、毎年度、国民年金・厚生年金の両勘定から「基礎年金給付費を被保険者数に応じて頭割りで按分した額」として基礎年金勘定に拠出される。この頭割り計算が基礎年金（bas）モジュールの核心的な役割である。

財政方式（積立方式と賦課方式の中間）

日本の年金財政は理念上の賦課方式・積立方式のいずれでもなく、両者の中間的な性格を持つ（厚生労働省, 2024a, 2024b）。厚生年金勘定・国民年金勘定はともに積立金を保有しているが、2004 年改正以降の財政方式の設計上、「100 年後に支出の 1 年分の積立金を保有する」水準が財政均衡の目安とされている。このため、積立金の推移（減少速度・枯渇時期）は財政検証の重要な出力指標となる。

マクロ経済スライドの仕組み

マクロ経済スライドは、年金改定率から「スライド調整率」（公的年金被保険者数の変動率＋平均余命の伸び率）を控除することで給付水準を自動的に調整する仕組みである（厚生労働省, 2024a, 2024b）。名目額の引き下げを行わない名目下限措置と、未調整分を翌年度以降に繰り越すキャリーオーバー制度により、デフレ期・低成長期には調整が先送りされる。

財政検証の計算上、このマクロ経済スライドの調整率（cuta）が bas で算出され、下流の emp_shushi に引き継がれて最終的な収支見通しに反映される。これが本プロジェクトの cuta パリティの議論（第 4.3 節）と直接つながる。

2.1 全体像：六つの計算ブロック

公式モデルは、おおまかに次の順で前提 → 給付 → 国年 → 基礎 → 収支 → 分布へと計算が進む。各ブロックは独立した実行ファイル（.exe）に相当し、Python 版では src/ 配下のモジュールに対応する。各ブロックの構成については表 1 に整理した。

イメージ: 上流で「世界の前提（外枠）」を置き、厚生・国年・基礎の各勘定を順に埋め、最後に厚生全体の収支表（01shushi 等）に集約する。プログラム上は中間 CSV を次のブロックが読む形だが、政策内容としては一つの試算シナリオの連鎖である。

厚生労働省資料との対応箇所は以下のとおりである（厚生労働省, 2024a）。ブロック 1（wakuc）は第 3 章第 4 節「経済前提の設定」・第 5 節「被保険者数の将来見通し」、ブロック 2（usys20）は第 6 節§1・§2、ブロック 3（nat）は第 3 節§2・第 6 節§2、ブロック 4（bas）は第 6 節§3、ブロック 5（asys20）は第 7 節§4「年金財政の将来見通し」、ブロック 6（bunpu）は第 5 章「年金額の分布推計」にそれぞれ対応する。

2.1.1 ブロック 1：外枠（wakuc）

「外枠」は、試算全体の共通前提パッケージである。将来の収支見通しは人口・経済・労働の前提に大きく左右されるため、wakuc がこの前提を試算し、後続の全モジュールが参照する。出力される主な量は以下のとおりである。

具体的には、国立社会保障・人口問題研究所の推計に準拠した年齢・性別別の将来人口推計をはじめ、厚生年金被保険者や第 1 号・第 3 号被保険者等の制度別・年齢別推計を算出する。また、年金制度の適用拡大の試算に直結するパート労働者の将来推移（simlpart.py）や、国家公務員・地方公務員・私学共済といった共済組合系加入者の動態（simlkyos.py）に関するミクロ的な動きもここで推計される。さらに、賃金や物価指数の将来推移といったマクロ経済前提もこのモジュールで生成され、試算全体の屋台骨を形成する。

2024 年財政検証では成長型経済移行・継続ケース、過去 30 年投影ケース等の複数の経済シナリオが想定されているが、これらの違いは wakuc 段階での経済前提パラメータの違いとして入力される。なお、各処理段階と対応する Python ファイルの一覧を表 2 に示す。

2.1.2 ブロック 2：厚生年金・給付費推計（usys20 / emp_kyufu）

「給付費推計」は、財政検証の最大かつ最も複雑なブロックである。厚生労働省資料第 3 章第 6 節§1・§2（「被保険者の区分、報酬及び保険料の推計方法」「受給者数及び給付水準を維持した場合の給付費の推計方法」）に対応し、以下を推計する。

本モジュールでは、まず年齢・性別・種別ごとの標準報酬額や加入期間の分布に基づき、被保険者の将来像を推計する。その上で、老齢・遺族・障害といった各受給事由について、過去の加入履歴を反映した年齢コーホートごとの積み上げ計算を実行し、受給者数の将来推移を導き出す。最終的に、給付水準を現行のまま維持したと仮定した場合の粗ベースでの給付費の将来推計や、マクロ経済スライド適用前の報酬比例部分の給付額が算出され、後続の収支計算モジュールへと引き継がれる。

年金受給は過去の加入履歴の蓄積であるため、現役世代の状況が将来の給付に反映されるまでに数十年かかる。このタイムラグを年齢コーホート別に積み上げる計算（simlbzw.py、simlrhnf.py）が数値処理の大部分を占め、単体完走に数分を要する。

重要な実装制約: main.py は初期化（zero_init()）をループ前に 1 回だけ呼ぶ設計のため、複数の pseid（制度）を同一プロセスで連続実行すると前制度の状態が配列に残留する。フルパリティの確認時は制度ごとに別プロセス起動（USYS20_PSEID_LIST=4 等の環境変数）が必須となる。なお、各処理段階と対応する Python ファイルの一覧を表 3 に示す。

2.1.3 ブロック 3：国民年金（nat）

「国民年金」は、第 1 号被保険者（自営業者・農業従事者・学生・無職等）を主な対象とした勘定の推計を担う。厚生労働省資料第 3 章第 3 節§2「国民年金」および第 6 節§2 が対応箇所となる。推計する主な量は、第 1 号被保険者数・納付率・免除率・受給者数（老齢基礎・障害基礎・遺族基礎）・国庫負担額である。

FP 累積誤差が顕著なモジュール: nat は約 105 年分の年次ループ（2022〜2120 年）を回し、各年で漸化式的な計算を繰り返す。C 言語の FMA 命令最適化と Python の厳格な IEEE754 評価の差が累積した結果、最大約 2.36%の相対誤差が高齢・終期のセルに集中する（第 5.1 節および表 4 参照）。

2.1.4 ブロック 4：基礎年金（bas）

「基礎年金」ブロックは、全制度共通の基礎年金給付費の計算と、各勘定への拠出金（頭割り）の配分計算を担う。計算上の中心的な役割は以下の 3 点である。

1. 基礎年金拠出金の頭割り計算 (atamawari.py): 基礎年金給付費を各制度の被保険者数に応じて按分
2. マクロ経済スライド改定率（cuta）の算出: スライド調整率を計算し、下流に引き渡す
3. 国庫負担の計算: 給付費の 2 分の 1 に相当する国庫負担額の算定

意図的なパリティ見送り: Bas モジュールでは、Python 版と R 版（参照データ）の間に意図的なパリティ見送りが存在する。Python 版は C ソースの意図を正確に反映しているが、R 版が独自の前提（計算終了年の延長・pre_cut 初期値の差異）を持つため、2025 年以降の cuta および関連列を compare の exclude として固定している。cuta は下流の emp_shushi にも伝播するため、exclude の範囲は収支見通しにも及ぶ。具体的な出力項目は表 5 に整理した。

2.1.5 ブロック 5：厚生年金・収支（asys20 / emp_shushi）

「収支計算」は財政検証の最終集計段であり、厚生労働省資料第 3 章第 7 節§4「年金財政の将来見通し」の出力に直接対応する。前段 4 ブロックの計算結果を入力として受け取り、年度ごとの収支表（01shushi.*.csv）を生成する。計算する主な量は次のとおりである。

収支計算の骨格は、収入と支出の差額を積立金に加減算していく動的な更新プロセスである。まず、保険料収入、運用収入、国庫負担、および各種支援金収入を合算して「収入計」を求める。一方、下流へ支払う基礎年金拠出金や、厚生年金独自の給付、ならびに管理費等を合算して「支出計」とする。この収入計と支出計の差分が「収支差」となる。そして、前年末の積立金に対して実質利回りを乗じて得られる「運用収入」を加味し、前年末積立金に収支差を加えることで、その年度の「年度末積立金」が確定する。この一連のサイクルを 100 年以上にわたって繰り返すことで、長期の財政見通しが構築される。

emp_shushi は上流 4 ブロックの出力をすべて入力として読むため、emp_shushi の長期収支が参照データと一致していることは、パイプライン全体の整合性を間接的に示す「統合テスト」に相当する。一方、compare_outputs.py だけでは長期収支を十分に検証できない場合がある（PARITY_OVERVIEW_rev2024.md 参照）。

- 盲点: 01shushiCSV では同一年度キーが収支見通しより前の別ブロックに先に出現すると、キー比較は先頭行のみを見るため、収支見通しブロックが compare 対象から外れうる。
- 補完: parity/plot_emp_shushi_parity.py（5 制度合算・「収支見通し【スライド調整後】」）および parity/check_01shushi.py 等で長期系列を直接比較する（第 5.2 節）。
- golden の注意: 現行 output/golden/fpp_r/.../01shushi では、スライド調整前の収入計が k≥25 で欠損（nan） となる世代がある。長期乖離の厳密比較では old_output/golden 等、数値が揃った参照を baseline にする運用がある。関連する検証用スクリプト一覧を表 6 に示す。

2.1.6 ブロック 6：分布（bunpu）

「分布」は、厚生労働省(2024a)の第 5 章「年金額の分布推計」に対応する。被保険者の標準報酬分布や受給者の年金額分布を計算し、所得代替率等の分布ベースの政策指標を導出する。2024 年財政検証では、性別・生年度別の老齢年金月額の分布（5 万円未満〜25 万円以上の区分別割合）と平均年金月額が公表されている (厚生労働省, 2024a)。財政検証の主要な収支指標（積立金・所得代替率の代表値）は上流で計算済みであり、bunpu はその内訳・分布の詳細化を担う補助的なブロックである。Python 版では標準シナリオにおける優先度は相対的に低く、比較設定の整備は次フェーズの課題としている。

2.2 厚生年金まわりの「制度別の枝」

厚生給付・収支では、試算モデル内部で被保険者区分ごとに別ファイル・別計算枝が用意されている。出力ファイル名の接尾辞（kou, kok, ren, sig など）と、内部 ID（pseid）で識別される（表 7 参照）。なお、定義のあいまいさに注意すべき点がある。ドキュメントや可視化スクリプトに出てくる「5 制度合算」は、国民年金・厚生・国保・地方共済・私学を法令上五つに並べた名称ではない。主に kok / kou / ren / sig / tou の五本の出力を年度で足し合わせたものである。接尾辞 kok を「国民健康保険（国保）」と読む表現が一部資料にあるが、本プロジェクトでは正式な制度対応は確定していない（国年改定率ファイル KOKUKAITE など、「国」系データとの関連はコード上ある）。制度名の正本は、元の C 仕様・業務資料との照合が必要である。

2.3 収支（shushi）の内容

収支とは、ここでは emp_shushi が出力する 2.2 節で述べた 5 制度の長期の財政見通し表（例: 01shushi.*.csv の「収支見通し【スライド調整後】」）を指す。年度ごとの収入計・支出計・収支差・積立金などが並び、財政検証の結果を一望するための集計面である。

- ミクロ的な一致: 各 CSV のセルを参照出力と一つずつ比較する（後述）。
- マクロ的な一致: 上記の収支系列を年度軸で比較し、平均・最大乖離率を見る（本報告第 5 節・図）。

収支は、前段（給付推計・基礎年金・国年・外枠）の結果を入力として読み、収支計算ロジックで再集計したものである。途中で無関係な数値をつぎはぎして収支だけ合わせているわけではない（compare の許容設定は「判定の閾値」であり、実行時に出力を書き換えるものではない）。

2.3.1 収支が長期に広がるメカニズム

積立金は漸化式的に更新される。支出計にわずかな差があると、毎年の収支差に乗り、積立金の水準差として蓄積し、翌年以降の運用収入（積立金×利回り）にも影響する。この複利的な増幅により、初期の微小差が 100 年超の試算で終期付近の大きな乖離として現れる場合が

ある。これは「ロジックが間違っている」のではなく、同一ロジックで生じた微小差の動学的増幅として理解すべきである（詳細は第 5.3 節）。

2.3.2 先行研究との接続：プログラム再現の先行事例

髙橋他 (2023) は 2019 年財政検証プログラムを Windows 環境で再現する際に以下の課題を報告している。コンパイラの変更に伴うコマンド・関数の修正が必要となったこと、一部のオプション試算では正確な再現が困難だったこと、再現可能なシナリオでも公表結果と最大 1/100 程度の差が生じたことである。

本プロジェクトは、こうした「バイナリ依存・環境依存」の問題を根本から解消することを目的としている。Python による完全再実装により、コンパイラ依存・実行環境依存の問題がなくなり、透明性と再現性が大幅に向上する。また、髙橋他 (2023)が報告した「再現誤差（最大 1/100 程度）」は、本プロジェクトで取り組んでいる浮動小数点パリティ問題（第 5 節）と本質的に同じ課題であり、先行研究の知見を引き継いでいる。さらに、同論文が実施した経済前提の感度分析（経済変動に対する所得代替率の安定性の検証）は、Python 化によって反復実行・可視化が容易になることで、より体系的に実施できる分析の一例でもある。

2.4 レガシー構成の課題

旧環境では、一部のコードを修正するたびにモジュールごとの再コンパイルが必要であり、さらにモジュール間が膨大な CSV ファイルの I/O に依存していたため、デバッグ作業が極めて困難であった。また、特定の C / C++ コンパイラや Windows 環境に依存したバイナリファイルで実行されるため、他環境への移植性や再現性に乏しいという弱点があった。加えて、そのような硬直化したパイプラインでは、分析に必須となる結果の可視化や、前提パラメータを変化させた反復試算、あるいは外部のミクロ経済モデルとの連携といった拡張対応が容易ではなかった。本プロジェクトにおける Python 化は、これらすべての制約を緩和し、現代的なデータ分析基盤へと移行させるための必須のプロセスである。

2.5 三系統の対照（公式 C・R・Python）— 作成者向け

財政検証の元のプログラム（C/C++ 実行ファイル）、財務省側で整備された R（R による一括実行＋従来 exe）、本プロジェクトの Python（Python フォルダ） は、同じ試算の別実装である。以下は「どの名前がどこに対応するか」を一覧にしたものである（パスは標準配置の例。環境により R のルートは異なりうる）。

2.5.1 実行単位（モジュール）の対応

旧環境の C 言語モジュールと Python 実装のブロックごとの対応関係を表 8 に示す。読み替えの要点は以下の通りにまとめられる。

- Python では emp_kyufu＝usys20、emp_shushi＝asys20 とフォルダ名だけが変わっている。ログや環境変数は USYS20_*（給付）、KOUSEI_*（収支）が残る。
- R は計算ロジックの実装ではない。fp_run01_*.R が 制御ファイル（*_CP.txt）を渡して exe を順に起動し、結果を suuri/rev2024/... に溜めるオーケストレーションである。

2.5.2 実行順序の対応（注意：三つとも同一ではない）

三系統（公式 C、R、Python）における実行順序の対応を表 9 に整理した。作成者向けの注意: 順序が違っても、各ブロックが読む CSV のパスと試算番号（例: 1000-3001-1000）が揃っていれば個別モジュールの突合は可能である。ただし 収支（asys20 / emp_shushi）は給付（shus.*）・基礎（cuta 等）・国年改定率を入力にするため、実務上は 給付・基礎・国年の出力が存在した後に収支を回す。

2.5.3 データ置き場の対応（suuri/rev2024 ツリー）

R / 公式 exe が慣習的に読み書きする R/suuri/rev2024/ と、Python が rev2024/output/（および読取専用の rev2024/data/）に置くものの対応例を示す。ゴールデン比較では、R 側ツリーを rev2024/output/golden/fpp_r/suuri/rev2024/ にミラーする（parity/import_r_golden.py）。各環境におけるデータの入出力と compare 対象 ID の対応については表 10 を参照のこと。ファイル

名の 1000-3001-1000 は、試算番号・経済前提番号・外枠番号の代表例（R 変数 ShisanNum, EconNum, 外枠番号）に対応する。

2.5.4 制御・起動パラメータの対応

各環境（R 版、Python 版）における制御・起動パラメータの対応を表 11 に示す。詳細な値の一覧は docs/MODULE_PARITY.md 内の §A「管制の標準」を参照。

2.5.5 usys20（厚生給付）内部：C ソースと .py の対応

usys20.exe の処理の骨格は、sepsd（推計本体） が年度・種別ループを回す構造である。Python では src/emp_kyufu/sepsd.py が同順序でモジュールを呼ぶ。C ソースと Python モジュールの詳細な対応関係は表 12 の通りである。

制度別実行: C / Python とも pseid ∈ {0,1,4,5}（kou / kok / ren / sig）ごとに給付推計を回し、出力ファイル名の接尾辞 _kou, _kok, _ren, _sig に反映する（§2.2）。並列実行は Python のみ USYS20_PARALLEL=1 でオプションあり。

2.5.6 asys20（厚生収支）内部：C ソースと .py の対応

収支プログラム（asys20.exe）は、給付側の shus.*.csv（usys20 出力）を読み、基礎・国年・外枠の各種 CSV を組み合わせて 01shushi（長期収支） 等を書く。収支側の詳細な構成については表 13 を参照。

収支側の制度インデックス: fopn.py では ifp10_usys[i] が i=1..4 → kou, kok, ren, sig（tou は usys20 が出さない場合のフォールバックあり）。出力ファイル名の _08kok 等は、この i と接尾辞の対応で追跡する。

2.5.7 その他モジュール（nat / bas / wakuc）の Python 配置

その他の主要モジュールにおける C 言語ソースと Python の配置の対応を表 14 に整理した。

2.5.8 本対照表の使い方（作成者向け）

実務において R 側の出力に疑義が生じた際は、まず R/suuri/rev2024/... のパスが上表のどのモジュールに該当するかを特定する。次に、対応する output/... ディレクトリ下の出力を Python パイプラインで再生成し、compare_outputs.py による差分比較を実行することで問題箇所を切り分ける。一方、Python 側の実装不備を修正する場合は、表内の C ソース列（prog/usys20/... など）と Python ソース列を直接突き合わせ、元の .cpp の挙動を正解として Python コードを書き換えるアプローチをとる。また、長期収支に乖離が見られた場合の調査手順としては、ミクロな数値差は 01shushi.*.csv で確認し、マクロな推移は parity/plot_emp_shushi_parity.py で可視化する。原因が上流にある場合は、給付モジュール出力の shus.* から収支モジュール入力の rdfl や shus へとデータの流れを遡って追跡することが基本となる。

第 3 節： R 版と Python 版の関係

3.1 三層の関係

財政検証システムの三層構造における本プロジェクトの位置づけを表 15 に示す。Python 版は R スクリプトの単純な翻訳ではない。C / C++ を読み解き、モジュール境界を保ちつつ Python で再構築している。R の出力は 「正解」ではなく参照データ として置き、差分が出たときは C と Python 実装のどちらが妥当か を切り分ける。

3.2 関連研究・技術動向

レガシー数値計算の Python 化、浮動小数点の検証、NumPy / Numba 等による高速化、年金財政のシミュレーション研究は、いずれも本件の文脈と接続する (Bagari, 2026; Cazzola and Favalli, 2024; Challa et al., 2025; Kukreja et al., 2025; Nguyen and Marché, 2011; Okamoto, 2013; Orvalho and Kwiatkowska, 2025; Pietrini et al., 2024; Shah, 2024; Suleiman et al., 2024; Sun, 2024; Zarudnyi and Koval, 2024; 山本, 2010; 橘木他, 2006; 蓮見・中田, 2009; 鈴木他, 2003)。

第 4 節：移植の進め方と現在の状況

4.1 標準的な実行の流れ

R 及び Python の各版における作業フローは以下のとおりである。

1. （任意・検証用） R 側で参照 CSV を更新 → output/golden/... に取り込み。
2. Python パイプライン: wakuc → bas → emp_kyufu → emp_shushi → nat（標準シナリオの順序は run_tests.bat / run_full_pipeline.bat に準拠）。
3. 比較: parity/compare_outputs.py が golden と actual の CSV を突き合わせ、レポートを output/compare_report/ に出力。

日常の再現実行は Python のみ に寄せる方針であり、参照更新は 必要なときだけ 行う（詳細は docs/Next_CleanUP.md［見出し「パリティ一段落後の作業」］）。

4.2 検証の二つのルート

ゴールデンデータとの検証には、ミクロとマクロの二つのルートが存在する（表 16 参照）。重要な点としては、長期収支の乖離が小さいからといって、モデル全体の組み込みがすべて終わったとは言えない。 マクロで近くても、ミクロでは 許容誤差付き pass・比較対象外（exclude）・未整理の TODO が残る場合がある。一段落の判定は compare ターゲット一覧と CURRENT_ISSUES.md に依る。

4.3 進捗サマリ

本プロジェクトの作業については、完全なパリティを達成しているわけではなく、かなり近いレベルまでの接近にとどまる。そのため、本作業はレベル 1（主要モジュールの構造と主要出力の実質整合）を目標にしている。docs/PARITY_OVERVIEW_rev2024.md・docs/CURRENT_ISSUES.md の整理と整合する要点は次のとおりであり、各処理における不一致の状況を表 17 にまとめた。

なお、表内の「判定」が「許容超過」となっている箇所については、既知の差異として適切に管理されている。例えば、wakuc における 1 セルの不一致は、浮動小数点の端数処理に起因するものであり、実質的な被保険者数（Ap）は全制度・全年度で完全一致しているため、

計算ロジック自体に誤りはない。また、bas モジュールの kekka ファイルで見られる約 21.5%（1,186 / 5,521 セル）の不一致（最大相対誤差 99.46%）は、マクロ経済スライドの調整終了年度（S_C_NENDO）が golden 側で 2039 年、Python 側で 2042 年と 3 年ずれていることに起因する既知の仕様差である。この不一致は終了年度以降の特定のカット率関連行に集中しており、基礎年金拠出金や年度末積立金といった主要な政策変数は 2023 年度時点で実質一致している。このため、自動比較ツール（compare）上では、設計意図に沿って kekka ファイルを除外した上で pass と判定する運用としている。各ブロックの進捗状況と初見向けの要約については表 18 を参照のこと。

メイン compare（3 本）: compare_config.json / compare_config_nat_skip3_check.json / compare_config_emp_hou.json が、ドキュメント上の最新記録では いずれも pass とされる時期がある（再実行のたびに要確認。正は output/compare_report/summary.json）。

このように自動比較においてあえて比較対象から外している意図的な仕様差は他にも存在する。マクロ経済スライド改定率（cuta）については、Bas 側で持っている初期前提（pre_cut）と参照データの間に差異があり、これを無理にコード修正で合わせるのではなく、仕様の違いとして exclude 指定で固定している。また、終了年度についても、Python 側は本来の C 言語の仕様に忠実な終了条件を採用しているのに対し、参照側の R プログラムでは終了判定が別年度まで延長されるケースがあり、この違いも比較上は既知の仕様差として取り扱っている。

4.4 実装漏れの典型

セル比較で差が大きかった事例の多くは、浮動小数点ではなく 移植漏れ・参照ずれであった（詳細は PARITY_DEBUG_GUIDE.md）。パリティ検証において直面した実装漏れの典型例と調査イメージを表 19 に整理した。

同種の疑いが残る箇所（siml / shke / dtst / seid の他ループ・読込・初期化）は、引き続き C 対照のチェックリストとして管理している。

4.5 最終監査における主要な移植漏れ課題

最新の AI ベースのコード監査（gemini_migration_completeness.md 等）により、長寿命の年次ループや特定の境界条件において顕在化する深部の実装漏れが特定されている。これらは現在残存しているミクロ・マクロ乖離の主要因と推定される。

第一に、厚生年金の集計処理（shkekiso）における「年齢ループ範囲の乖離」である。元のC++ 版では全年齢層（0 歳から 115 歳）を対象に基礎データ（okisor）の累積処理を行っているが、移植された Python 版ではループが 15 歳から 89 歳の範囲で打ち切られていた。これにより、若年層や 90 歳以上の超高齢層のデータが計算からごっそり欠落し、重大な数値乖離の温床となっていた。

第二に、「多次元配列の初期化漏れ」である。C++ 側の zero.cpp で行われていた巨大な 5 次元配列（os2, j2, j3 等）のゼロクリア処理が、Python 版では明示的に定義されていなかった。これにより、特定の条件下において未初期化の変数を参照してしまい、数値が非決定的に不安定化するリスクが内在していた。

第三に、細かな「条件分岐や定数定義の不備」である。例えば、simlrhnf モジュールにおいて、死亡や失権が発生した後の給付調整ロジック（特定の乗算係数の適用）が欠落していたり、flt 等の定数初期化値が C++ 版と異なっていたりするなど、コードの深部に潜む仕様違いが、長期試算における予期せぬズレを引き起こしていた。

これらの課題は、今後の最終的なパリティ完全一致に向けた最優先の改修事項として「CURRENT_ISSUES.md」にて一元管理されている。

第 5 節：精度・長期収支・性能

本作業では個々のセルの値を丁寧に突き合わせるミクロ検証と計数の集計値でかつ代表的な変数となる収支変数に関するマクロ検証の両面から作業を進めた。ミクロ計数からは制度的な実装の整合性を確認し、マクロ計数からは制度全体の統合状況や誤差の全体から見た誤差の発生状況を把握するアプローチとなっている。

5.1 ミクロ検証と浮動小数点（FP）

ミクロ検証の過程では、浮動小数点（FP）の計算精度や丸め誤差が課題となる局面も多かった。例えば、emp_kyufu モジュールの hou2 出力においては、入力データ（dtst）の先頭 4 行の読み込みスキップ処理を修正した結果、セル不一致が完全にゼロとなる完全再現を達成した。一方で、nat モジュールの一部出力においては、データ読み込み直後の初期状態では値が一致するものの、100 年以上にわたる年次ループを繰り返す中で、浮動小数点の誤差が徐々に

累積し、終期において数パーセントの相対差として顕在化するケースがあった。これはプログラムの論理バグではなく、C 言語（コンパイラの FMA 最適化等）と Python（厳格な IEEE754 評価）間の演算順序や丸め誤差の違いによる不可避な差異である。そのため、この種の差異は許容誤差設定（per_file_tolerance）を設けて管理する運用としている（詳細は PARITY_DEBUG_GUIDE.md 付録 FP を参照）。

なお、C 言語と Python の浮動小数点演算の差異は、FMA（Fused Multiply-Add）命令の使用有無に起因する。通常の積和演算 $a + b \times c$ は、乗算・丸め・加算・丸めという 2 段階の丸めを経る。Python はこの IEEE754 規格どおりの 2 段階評価を厳格に行う。一方、C 言語のコンパイラは最適化の過程で FMA 命令を使用することがある。FMA は乗算後の中間結果を丸めずにそのまま加算まで一気に実行するため、丸め回数が 1 回少なく、数学的な正確値により近い結果を返す。

この 1 演算あたりの微小な差（10^{-16}程度）は単独では無視できるが、nat モジュールでは老齢年金受給者数の更新を約 100 年分・年齢 116 階級にわたって繰り返す漸化式計算を行うため、差が年々引き継がれながら蓄積する。高齢者ほど・試算終期ほど多くのループを経た値であるため、誤差は高齢・終期のセルに集中し、最大で約 2.36%の相対誤差として観察される。

なお、論理エラーであれば誤差は常に一方向に生じるが、FP 累積誤差の場合は演算ごとに C 版が大きかったり Python 版が大きかったりするため、年度をまたいで誤差の符号が反転する。本プロジェクトでもこの符号反転が確認されており、「計算ロジックは同一であり、差異は浮動小数点の累積的なドリフトに起因する」と判定する根拠となっている。また、C 言語の FMA 使用はコンパイラ・最適化フラグ・CPU 世代によって変わるため、同じ C ソースでも環境によって結果が異なりうるという問題を抱えている。その点を考えると、Python による再現性を優先した議論は重要であろう。

5.2 長期収支見通し（マクロ検証）

マクロ検証では emp_shushi の 01shushi（スライド調整後）、試算モデル内の kok / kou / ren / sig / tou を年度で合算した系列（--source 01shushi_sum、2024–2120 年・97 点）を対象としている。なお、数値は docs/emp_shushi_parity_stats.csv（parity/plot_emp_shushi_parity.py 再生成時に更新）にある。

5.2.1 乖離率統計（フルパイプライン実行後）

マクロ指標（厚生年金収支等）の平均および最大乖離率の統計結果を表 20 に示す。

5.2.1a モジュール別ミクロ検証（§2.5.1 実行単位との対応）

§2.5.1 の六ブロックに対し、parity/parity_quantify.py が golden と actual の 数値セルを突き合わせた結果を集計した（生成: docs/PARITY_QUANTIFICATION.md、2026-05-22 時点の output/）。齟齬率は strict 不一致セル数 ÷ 比較対象セル総数（compare 運用許容より厳しい判定）。emp_kyufu は u-rev と hou2 の合算、nat は出力種別 4 行の合算である。政策ブロック別ミクロ検証の結果は表 21 にまとめた。

bas モジュールの給付水準調整結果における差異について

基礎年金を試算する bas モジュール（基礎年金拠出金の按分とマクロ経済スライドの算出を担う計算ブロック）の出力のうち、給付水準調整結果ファイル（kekka） について、Python で再実装した計算結果と golden（従来の C 言語プログラムを R 経由で実行して得た参照用の出力データ）を、表形式データのセル単位で突き合わせたところ、比較対象となったセルの約 21.5% で数値が一致しなかった。なお、同一モジュールの別出力である積立金見通しファイル（TUMATUMI）では不一致はゼロであり、21.5% という比率は kekka の 2 ファイルに限った現象 である。本節では、その意味と取り扱いを、プログラム実装の詳細に立ち入らずに整理する。

不一致の直接原因は、マクロ経済スライドの調整終了年度（内部変数 S_C_NENDO：スライド調整をいつまで適用するかを示す年度）が、参照データ側と Python 実装側で 1 年ずれていることにある。具体的には、golden（FPP_R 由来）では終了年度が 2039 年、Python 実装では 2040 年 となっている。この 1 年の差により、2025 年度以降に適用される スライド改定率（cuta）が、どの年度の累積調整結果を参照するかという点で食い違い、kekka 内の 2025 年度以降の行・列が一括して不一致として検出される。見かけ上の不一致率が大きいのは、計算全体が誤っているのではなく、終了年度以降のスライド関連の行が、別の前提で並んでいるためである。

S_C_NENDO は、給付水準調整（プログラム内部では tyousei と呼ぶ処理）が、次の考え方に沿って年度ごとにループを進めた結果として決まる。「年金財政の積立金が、支出の一定倍率（所定の水準）を下回った状態が続く限りスライド調整を継続し、初めて その閾値以上に回復した年度を、調整終了年度とする」という終了条件である。golden と Python 実装は、い

ずれも公式の C プログラムと同型のロジックを用いるが、途中の積立金や支出額などに、ごくわずかな数値差が生じる。財政検証のように 100 年超の長期試算では、その差が年を重ねるごとに蓄積し、終了判定が行われる年度が 1 年だけ前後する ことがあり得る。したがって、ここで観測されている 1 年差は、計算ロジックの誤り というより、長期シミュレーション特有の数値計算上の差異 として理解するのが妥当である。

一方で、年金財政の議論で重視されやすい 主要な政策変数 は、golden と実質的に一致している。kekka における不一致は、終了年度やスライド改定率に紐づく特定の行・列に集中しており、試算の中核となる水準指標は大きくずれていない。たとえば 2023 年度 の 基礎年金拠出金（各制度から基礎年金勘定へ拠出する金額）は、golden で約 3.0×10^{12} 円、Python で約 3.0×10^{12} 円である。2023 年度末積立金 も、golden で約 1.366×10^{13} 円、Python で約 1.365×10^{13} 円と、いずれも同桁・近い水準である。また TUMATUMI ファイル（積立の見通しをまとめた出力）では、777 セルを比較して 不一致 0 セル であり、こちらは完全一致している。基礎年金モジュールにおける主要指標の比較結果を表 22 に示す。

この差異は、本プロジェクトでは既知の仕様差として管理している。出力を自動比較する compare ツール（compare_outputs.py）では、kekka を比較対象から除外し、基礎年金モジュール全体の判定は pass（合格）としている。除外の根拠は次のとおりである。第一に、S_C_NENDO の 1 年差は、R 側の長期計算における収束のされ方に伴うものであり、Python 実装の誤りとみなさないこと。第二に、政策判断の材料として重要な 2023 年度前後の主要変数は一致していること。第三に、kekka の差は 2025 年度以降 のスライド改定率の参照の違いにとどまり、短中期の制度設計や収支見通しの読み取りに直結しないことである。

今後の対応としては、次の三つの方向が考えられる。第一に、golden を最新の R 実行結果で再生成し、終了年度の差が縮小するかを確認する方法である。第二に、現行の compare 設定（kekka の exclude・skip）を維持し、本節の説明をもって 仕様差として確定 する方法である。第三に、tyousei の終了条件や中間量の突合をさらに精緻化し、S_C_NENDO を golden と一致させる方法だが、これは中長期の技術課題として位置づけるのが現実的である。現時点では第二の方針を採用しており、compare ツール上は bas モジュールを含むパイプライン全体が pass を示している。報告書の表で基礎年金の齟齬率が 21.5% と示される場合は、厳密な全セル比較の参考値であり、運用上の合格判定とは別指標である点に留意してほしい。

読み分けについては、上表は CSV セル単位のミクロである。emp_shushi の長期収支（01shushi 合算）は compare のキー比較では収支見通しブロックが漏れうるため（§ 2.1.5）、次節のマクロ表（§ 5.2.1 上表・§ 5.2.1b）と併読するのが望ましい。

5.2.1b 最大乖離率一覧（§2.5.1 単位）

ミクロは各モジュール compare 出力のセル最大相対誤差（strict 不一致セルについて max(|g−a|/max(|g|,|a|))）。マクロは §5.2.1 の 5 制度合算・スライド調整後系列である。ブロック別の最大乖離率一覧を表 23 に整理した。

再集計コマンド（数値更新時）:

```
py -3 parity/parity_quantify.py
py -3 parity/aggregate_parity_by_module.py
py -3 parity/plot_emp_shushi_parity.py --source 01shushi_sum
```

再集計された指標を読み解くと、まず「総報酬」や「保険料」といった上流の計算結果は参照データと実質的に一致しており、給付推計の入力となる収入経路のロジックは高い精度で揃っていることが確認できる。「収入計」については、全体平均で約 2%、終期の 2120 年時点で最大約 6.5% の乖離にとどまっている。これは、政策判断において「将来見通しがおおむね同じトレンドを描いているか」を確認するマクロ指標としては十分に有用な精度であるが、セル単位の完全一致を保証するものではない点には留意が必要である（第 4.2 節の留保事項）。また、「収入計」や「積立金」の乖離率は、直前のプロット世代（収入計平均 4.45%、最大 13.07%）から着実に改善を遂げており、過去の記録最良値（3.25% / 10.14%）に近い水準まで回復している。一方で「収支差」やそれに伴う「積立金」に関しては、毎年の支出側に生じるわずかな計算差が、利回り計算を通じて積立金へ複利的に蓄積されるため、試算の終盤になるほど乖離が増幅しやすいという構造的な特徴を持っている（詳細は第 5.3 節を参照）。

5.2.2 改善の推移（同一指標・報告用）

docs/MODULE_PARITY.md §C・docs/REPORT_MATERIAL.md §8 と同型の整理。収入計（5 制度合算・スライド調整後）のみ抜粋したパリティ改善の推移については表 24 に示している。

報告での言い分け（REPORT_MATERIAL.md §8.3 と整合）における状況区分の基準については表 25 に示す。

再生成コマンド（図・CSV 更新時）:

py -3 parity/plot_emp_shushi_parity.py --source 01shushi_sum

このスクリプトによる集計変数の時系列的推移の水準については図 1 を、乖離率の推移については図 2 をそれぞれ参照のこと。

5.2.3 2040 年における乖離率の急拡大について

上記の「集計変数の時系列的推移（乖離率）」のグラフにおいて、2040 年を境に収支差や積立金の乖離率が急激に拡大する現象が見られる。これは、前述したマクロ経済スライドの調整終了年度（S_C_NENDO）の 1 年のズレがマクロ指標に直接波及した結果である。

第一の要因は「給付抑制の 3 年延長」である。参照用の golden データ（FPP_R）ではマクロ経済スライドによる給付水準の抑制が 2039 年に終了するため、2040 年以降は抑制が行われない。一方、Python 実装では終了判定が 3 年遅れて 2042 年となるため、この年において Python 側のみ 3 年分余計に給付水準が抑制（カット）されることになる。マクロ経済スライドの調整終了年度（S_C_NENDO）は、基礎年金モジュールの tyousei 関数が決定する。終了判定のロジックは golden（FPP_R）と Python 実装で完全に同一であり、境界条件（<）も一致している。S_C_NEDO が golden（2039 年）と Python（2042 年）で 3 年異なるのは、実装の誤りではなく、第二の要因として挙げられる上流モジュール（wakuc・bas）における 100 年超の長期シミュレーションで浮動小数点演算の微小な累積差が積立金・支出の推移に影響し、閾値到達年度が 3 年ずれた結果である。

第二の要因は、その 1 年の差が生む「ベースラインの恒久的なズレと複利効果」である。2040 年に Python 側で余分に給付が抑制された結果、同年の支出額は golden データに対して相対的に低く算出される。公的年金の給付額は前年の水準をベースに改定されていくため、この 2040 年に生じた支出のギャップは一時的なものではなく、それ以降のすべての年度において恒久的なベースラインのズレとして残存し続ける。さらに、この支出の差額（浮いた給付費）はそのまま当該年度の収支差のプラス分となり、積立金に上乗せされて運用利回りの対象となる。年を追うごとにこの積立金差額が複利効果によって雪だるま式に増幅するため、2040 年を起点として乖離率のグラフが急角度で、あたかもワニの口のように開いていく時系列的な特徴が明確に表れているのである。

このように、グラフ上の 2040 年の特異点は、単なる累積誤差の爆発ではなく、「スライド終了年度の 1 年の差」という明確なロジックの違いがマクロ経済のダイナミクスを通じてマクロ指標に増幅・反映されたものである。

5.3 乖離が長期に広がりうるメカニズム（収支の構造）

収支表では、おおまかに 年度末積立金 ＝ 前年積立 ＋（収入計 − 支出計）、運用収入が積立と利回りに依存する。支出計の構成要素（例: 独自給付）が参照よりわずかに小さいと、収支差がプラスに振れ、それが積立に載り、100 年超の試算では終期付近で収入計・積立金の乖離として目立つことがある。これは「収支だけ別計算でつなぎ合わせた」ことではなく、同一パイプライン上の微小差の動学的増幅として理解する。調査整理（docs/HANDOFF_NEXT_SESSION.md）では、厚生（kou）の終期付近で次の連鎖が確定している。

具体的には、厚生年金（kou）における独自比例部分（約 −3%）と独自定額部分（約 −2%）のわずかな過小評価が、毎年の支出計を恒常的に約 2% 引き下げる。支出が減ることでその年の収支差はプラスに振れ、余剰分が積立金へと蓄積される。これが数十年にわたって繰り返されると、利回り計算による複利効果も相まって積立金の乖離は急速に拡大し、試算 125 年目（y125）には約 +48% もの差へと膨れ上がる。積立金が過大になればそこから得られる運用収入も連動して拡大するため、最終的には収入計そのものに約 +20% の乖離をもたらすという連鎖構造である。

compare 3 本は ALL PASS でも、上記は マクロ指標（§5.2）と制度別 01shushi 詳細で残る課題である。D3bxtp 等の上流系列は実質一致に近い一方、独自比例・独自定額の数%差が残差の主因候補として追跡中である。

5.4 性能（参考）

emp_kyufu 単体完走は数分オーダー。ループのベクトル化など局所最適化はあるが、全体は引き続き Numba / Cython 等の検討余地 がある（詳細は CURRENT_ISSUES.md 参照）。

第 6 節：結論と展望

本研究は、別プロジェクトで進行している「動学的マイクロシミュレーション（Dynamic Microsimulation）」とのシームレスな接続を目指して、制度側からのプログラミング開発を行っている。動学的マイクロシミュレーションによって精緻に推計された個人のキャリアパスや「適用拡大対象者の属性（具体的な報酬水準、被保険者期間、非正規雇用の推移など）」の

膨大なミクロデータを、Pandas データフレーム等の形式で、第 2 部である本モデル（マクロ財政検証モデル）のインプットデータとして直接・動的に組み込むことが可能になる。その状況を踏まえ、以下のように総括と展望を述べる。

6.1 総括

本研究における最大の成果は、財政検証を構成する六つの主要計算ブロックがすべて src/ 配下に移植され、標準的な経済シナリオであれば Python だけで一貫してシミュレーションを実行できる段階に到達したことである。計算の信頼性については、ミクロレベルでの全セル比較（compare）によって実装漏れやデータの読み込みズレを徹底的に潰しつつ、マクロレベル（01shushi_sum プロット等）で長期収支の動態を俯瞰的に確認する、という双眼的なアプローチが有効に機能している。

長期収支の現状を見ると、最新のプロット結果では収入計の乖離率が平均 1.98%、最大 6.48% となり、過去の記録最良値（3.25% / 10.14%）を大幅に上回る水準まで改善した。終期における収支差や積立金の乖離も 15% 前後まで縮小しており、実用上のマクロトレンドの再現性は極めて高いと言える。ただし、ミクロ比較が「pass」と判定されたからといって、すべてのセルや系列が参照データと完全に一致したと錯覚してはならない。bas モジュールの kekka ファイルにおける意図的な除外、許容誤差（tolerance）に依存したターゲット、あるいは 01shushi におけるキー比較の盲点といった「未完了の含意」は依然として存在しており、これらは引き続き docs/CURRENT_ISSUES.md にて技術負債として厳密に管理していく必要がある。

6.2 展望と今後の運用方針

パリティ検証が一定の収束を見せた今、次なるステップは開発環境の整理と改善である。マクロ収支が十分な一致レベルに到達したことを契機に、R によるレガシーな参照環境と、Python による新たな本番環境の運用を明確に分離する方針をとる。今後の日常的な試算業務は Python パイプラインのみで完結させ、R 環境は前提条件の大幅な変更など、どうしてもゴールデンデータを再生成する必要が生じた場合にのみ稼働させる。また、開発過程で各所に分散していた技術的な課題やデバッグの記録は、すべて CURRENT_ISSUES.md に集約して一元管理する。これにより、同ドキュメントが今後の運用保守における唯一の「Runbook（運用手順書）」兼「残課題リスト」として機能することになる。

さらに、C 言語から Python へとコードベースが刷新され、可読性が劇的に向上したことの恩恵は計り知れない。今後の法改正や制度変更に伴うロジック改修において、プログラミング支援型 AI などを活用しやすくなり、モデル修正の効率と安全性が飛躍的に向上すると期待される。そして中長期的な最大の展望は、動学的マイクロシミュレーションとの連携である。これにより、短時間労働者への適用拡大などのミクロな制度変更が、マクロな年金収支やマクロ経済スライドの調整期間に与える影響を、従来のような簡便な補正ではなく、ボトムアップで正確に算出できる画期的な道が開ける。今後は、両モデル間で効率的に受け渡し可能なデータフォーマット（Parquet 等）を整備し、ミクロモデルとマクロモデルを完全に統合した、次世代の統合年金収支計算シミュレータを実現することが中核的な研究課題となる。今後、年金制度の適用拡大の対象者など、個人レベルの多様な属性やキャリア軌跡を精緻にシミュレートしたミクロデータを、本 Python マクロモデルのインプットとして動的に接続することで、より解像度が高く機動的な「次世代統合シミュレーションモデル」の構築へとつながる研究課題に拡張してゆく予定である。

付録：リポジトリ内の参照ドキュメント

リポジトリ内に存在する主要な参照ドキュメントとその用途の一覧を表 26 に示す。

図表

表 1 旧環境（R/C 言語）と Python 版のレイヤ構成比較

順序	従来	Python	政策・数理上の役割
1	wakuc	wakuc	外枠: 人口・労働・経済成長・物価など、後段共通のマクロ前提
2	usys20	emp_kyufu	厚生年金・給付費: 被用者年金将来給付・加入動態
3	nat	nat	国民年金: 第 1 号等の推計、納付・給付関連の勘定
4	bas	bas	基礎年金: 基礎年金部分の収支・各制度からの拠出
5	asys20	emp_shushi	厚生年金・収支: 保険料・給付・積立・スライド調整後の長期収支見通し
6	bunpu	bunpu	分布: 報酬・年金額の分布（所得代替率等の材料）

表 2 移行対象の主要ファイルと対応する C 言語ソース

ファイル	対応する C ソース	処理内容
main.py	main.cpp	エントリ・制御
simlpart.py	simlpart.cpp	パート労働者の将来推計
simlkyos.py	simlkyos.cpp	共済組合加入者の将来推計
file_io.py	fileio.cpp	外枠 CSV の入出力

表 3 各処理段階と対応する Python ファイル

段階	Python	C	内容
初期化	main.py	main.cpp, zero.cpp	エントリ・配列初期化
設定読込	cntl.py	cntl.cpp	試算番号・経済前提番号等
ファイル管理	fileio.py	fileio.cpp	ファイル開閉・パス管理
外枠読込	waku.py	waku.cpp	g.l, g.pop（被保険者数・人口の年齢別配列）
経済前提	econ.py	econ.cpp	賃金・物価・運用利回りの将来系列
制度パラメータ	seid.py	seid.cpp	保険料率・給付乗率等
基礎率	kiso.py	kiso.cpp	Kakudai_Ichibu 等の設定
初期データ	dtst.py	dtst.cpp	被保険者・受給者の初期実績データ読み込み
推計本体	sepsd.py	sepsd.cpp	年度×種別のループ制御
年度更新	siml.py	siml.cpp	給付シミュレーション
報酬比例給付	simlbzw.py	simlbzw.cpp	報酬比例給付の将来推計（90nenbe 等）
給付細目	simlrhnf.py	simlrhnf.cpp	老齢・遺族・障害給付の細目
財政再計算	simlsaite.py	simlsaite1/2.cpp	財政の再計算サブルーチン
集計	shke.py, pstat.py, stat_kyufu.py	shke.cpp, pstat.cpp, stat.cpp	給付費の集計・統計
中間出力	crshfl.py	crshfl.cpp	shus.*ファイル書き出し（収支モジュールへの引き渡し）

段階	Python	C	内容
法 2 給付出力	outhou.py	outhou.cpp	hou2.*ファイル

表 4 Python 版と C 言語版の実装の相違点

Python	C 系統	内容
dtst.py	dtst	被保険者・受給者初期データ
kiso.py	kiso	Kakudai_Ichibu（老齢一部給付乗算係数）・Shikyuritu_Kafu の設定
siml.py	siml	年次シミュレーション本体 （HIHO/ROREI/IZOKU/SHOGAI 等）
shke.py	shke	統計集計
printout.py	printout	CSV 出力（KISONENKIN, ROREI_SHINKI, DOKUZI 等）

表 5 Python 版で生成される主要な出力内容

Python	内容
atamawari.py	基礎年金拠出金の頭割り計算（被保険者数に応じた按分）
tyousei.py	マクロ経済スライド調整計算（cuta の算出）
read_hiyousha.py	被用者（第 2 号被保険者）数の読み込み
read_kokunen_jisseki.py	国年実績データ（NOUFU インデックス）の読み込み

表 6 検証・プロット用スクリプトと用途一覧

ファイル	処理内容
fopn.py	ファイル開閉・制度別パス管理 （ifp10_usys[i]で i=1..4→kou,kok,ren,sig）
econ.py	経済前提 CSV 読み込み
rdfl.py	shus.*ファイル→Cc, D3bxtp 等の配列に展開
shus.py / shus_calc.py	収入計・支出計・積立金の計算（Cc 配列）
shus_out.py	01shushi.*.csv の主要出力 （tanni[i]=1e-8 で単位変換）
shus_fullout.py	詳細系列の追加出力

表 7 外枠（wakuc）モジュールの主要な変数対応

接尾辞	試算モデル呼称	pseid	想定される対象
kou	厚年（一般）	0	厚生年金
kok	国共	1	国家公務員共済
ren	地共	4	地方公務員共済
sig	私学	5	私立学校教職員共済
tou	合算用	—	収支側の集計

表 8 実行単位ブロックの C 言語と Python の対応関係

ブロック	ファイル	C/C++の場所	R での起動	Python
外枠	wakuc.exe	prog/wakuc/ 等	fp_run01_20250228.R 内 shell("wakuc < wakuc_CP.txt ...")	src/wakuc/main.py
厚生・給付費	usys20.exe	R/prog/usys20/*.cpp（厚年 U-sys）	同 shell("usys20 < usys20_CP.txt ...")	src/emp_kyufu/main.py（内部名は usys20 のまま）
国民年金	nat.exe	prog/nat/ 等（nat.c 系）	shell("nat.exe ... infile outfile ...")（16 引数）	src/nat/main.py
基礎年金	bas.exe	prog/bas/ 等（main.c 系）	shell("bas.exe ...")	src/bas/main.py
厚生・収支	asys20.exe	R/prog/asys20/ 等（収支・kousei 系）	shell("asys20.exe < asys20_CP.txt")	src/emp_shushi/main.py（フォルダ名 emp_shushi、C の asys20/kousei に相当）
分布	bunpu.exe	prog/bunpu/（C++）	フル fp_run01 の後段	src/bunpu/main.py

表 9 スクリプト実行フローの段階別整理

系統	典型的な実行順	備考
公式 C の説明順	wakuc → usys20 → nat → bas → asys20 → bunpu	業務ドキュメント上の論理順に近い
R fp_run01_20250228.R（一部）	wakuc → usys20 → nat	ゴールデン生成の最小連鎖として docs で整備（MODULE_PARITY.md §A）
Python（標準）	wakuc → emp_kyufu → nat → bas → emp_shushi → bunpu	run_tests.bat 先頭コメントと一致
Python（検証用）	（任意）R 全 exe → golden 更新ののち、wakuc → bas → emp_kyufu → emp_shushi → nat	compare 用。順序は bat 定義どおり

表 10 R 版と Python 版の実行例と compare 対象 ID

内容	R 参照側（例）	Python 実行結果（例）	主な compare ターゲット id
外枠 CSV	suuri/rev2024/wakuc/rslt/ver_4_1/rslt1000/waku1000-*.csv	output/wakuc/rslt/ver_4_1/rslt1000/waku1000-*.csv	wakuc_waku1000_rslt_all
厚生給付 u-rev	suuri/rev2024/emp/rslt/u-rev/shus/shus.{試算}-{経済}-{外枠}_{kou,kok,ren,sig}.csv	output/emp/rslt/u-rev/shus/...（同名）	（収支の入力；単体 compare は shusg / kisor 等）
厚生 hou2	suuri/rev2024/emp/rslt/u-rev/hou/hou2.*.csv	output/emp/rslt/u-rev/hou/hou2.*.csv	compare_config_emp_hou.json
基礎 rslt	suuri/rev2024/bas/rslt/（cuta-*.csv, kekka*.csv 等）	output/bas/rslt/	bas_rslt_all, bas_kekka
国年出力	suuri/rev2024/nat/ 配下（KISONENKIN.csv, ROREI_SHINKI.csv 等）	output/nat/	nat_outputs_mapping
長期収支	suuri/rev2024/emp/rslt/ez_arev/shushi/01shushi.*_08{kou,kok,...}.csv	output/emp/rslt/ez_arev/shushi/01shushi.*_08*.csv	emp_shushi_ez_arev_shushi
カット率（収支）	suuri/rev2024/emp/rslt/ez_arev/cutr/cuta-*.csv, cutb-*.csv	output/emp/rslt/ez_arev/cutr/	emp_shushi_ez_arev_cutr
初期データ	suuri/rev2024/emp/data/, bas/data/, nat/data/ 等	data/（読取）	（compare 対象外）

表 11 三層構造（データ生成・モデル・可視化）の移行状況

項目	R / 公式（例）	Python（例）
試算番号	ShisanNum / CP 1 行目 → 1000	USYS20_INAME, Wakuc CP, g.Nfile
経済前提	EconNum → 3001	USYS20_IECON, g.Ecfile
外枠番号	usys20 CP 4 行目 → 1000	USYS20_IWNAME, g.Wcfile
厚生給付モード	UsysKey → 11	USYS20_KEY
国年 16 引数	nat_CP 並び	nat/main.py 引数省略時の自動注入
収支・予備（Cutr）	asys20_CP の Cutrfile4 等	g.Cutrfile4（非対話既定 000 等）

表 12 厚生年金給付費（emp_kyufu）の処理段階と対応ソース

処理段階（C++ sepsd.cpp 相当）	R ソース例	Python src/emp_kyufu/
エントリ・初期化	main.cpp, zero.cpp	main.py → zero_init, zero_sepsd
制御読込	cntl.cpp	cntl.py
推計本体	sepsd.cpp	sepsd.py
ファイル開閉	fileio.cpp	fileio.py
外枠読込	waku.cpp	waku.py（output/wakuc/.../waku*.csv）
経済前提	econ.cpp	econ.py
制度・率読込	seid.cpp	seid.py, seid_sik_expand.py
基礎率・加入	kiso.cpp	kiso.py
データ読込	dtst.cpp	dtst.py
集計（中間）	shke.cpp	shke.py
年度更新・給付シミュレーション	siml.cpp, simlg.cpp, simlbzw.cpp, simlrhnf.cpp	siml.py（内で simlg, simlbzw, simlsaite を呼ぶ）
所定・財・対（サブルーチン群）	simlsaite1.cpp, simlsaite2.cpp	simlsaite.py（saitezojuk 等）
その他シミュレーション補助	simlsaite, dsitk.cpp, rousaki.cpp	dsitk.py, rousaki.py
統計・脚下・比較表	pstat.cpp, stat.cpp, outhou.cpp	pstat.py, stat_kyufu.py, outhou.py
結果 CSV 整形	crshfl.cpp, outkn.cpp	crshfl.py, outkn.py
グローバル状態	各種ヘッダ・配列	state.py, constants.py

表 13 厚生年金収支（emp_shushi）の処理段階と Python 特有の差異

段階	Python emp_shushi/	主な入力（Python 側パス例）
エントリ・定数	main.py, set_constants.py	環境変数 KOUSEI_OUTPATH → output
対話制御・試算番号	cntl.py	非対話: KOUSEI_NONINTERACTIVE=1（run_emp_shushi_helper.py）
ファイル開閉	fopn.py（alfopen）	emp/rslt/u-rev/shus/shus.*_{kou,kok,ren,sig}、bas/rslt/cuta-*, nat/data/KOKUKAITE-* 等
経済前提	econ.py	emp/data/u-rev/econ/econ-{経済}.csv
給付 CSV 読込	rdfl.py	上記 shus.*（D3bxtp / 90nenbe 等の中間系列）
収支計算・スライド	shus.py, shus_calc.py	配列 Cc（収入計・支出計・積立金…）
詳細出力	shus_out.py, shus_fullout.py	emp/rslt/ez_arev/shushi/01shushi.*.csv

表 14 主要モジュールの C 言語関数と Python ファイルの対応

ブロック	C 側の主関数列（概念）	Python 主要ファイル
nat	cntl → file_open → seid → econ → waku → noufuritu → dtst → kiso → siml → shke → stat → printout	main.py, dtst.py, kiso.py, siml.py, shke.py, stat_calc.py, printout.py
bas	cntl → file_open → dtst → econ → waku → atamawari → tyousei → printout	main.py, atamawari.py, tyousei.py, read_hiyousha.py, read_kokunen_jisseki.py
wakuc	cntl → 外枠シミュレーション各種	main.py, simlpart.py, simlkyos.py, file_io.py

表 15 財政検証システムの三層構造と本プロジェクトの位置づけ

層	内容	本プロジェクトでの位置づけ
C / C++ ソース	公式に近い計算仕様の根拠	Python 移植の正とする仕様
R（R ラッパー＋従来 exe）	一連試算の自動実行と出力の蓄積	ゴールデン（参照 CSV）の生成に利用
Python	src/* の再実装	検証対象（actual）

表 16 ゴールデンデータ検証の二つのルート（ミクロとマクロ）

ルート	何を見るか	わかること
ミクロ	個別 CSV のセル単位の一致	実装漏れ・読込ずれ・初期化欠落の発見に有効
マクロ	長期収支（01shushi 等の収入計・積立金など）	試算全体のダイナミクスが参照に近いか

表 17 各処理における不一致の状況

モジュール	ファイル数	セル総数	strict 不一致	tol 不一致	最大相対誤差	判定
wakuc	100	4,616,343	1	1	0	許容超過 (1 ファイル)
bas	3	5,521	1,186	1,186	99.46%	許容超過 (2 ファイル)
emp_shushi	12	18,905	0	0	0	完全一致
emp_kyufu u-rev	17	22,528,898	0	0	0	完全一致
emp_kyufu hou2	4	282,240	0	0	0	完全一致
nat KISONENKIN	1	656,776	0	0	0	完全一致
nat ROREI_SHINKI	1	1,455,168	1,519	0	≤0.01%	許容内一致
nat DOKUZI	1	1,166	0	0	0	完全一致
nat（その他）	4	69,324	298	0	1.0	許容内一致
bunpu（一部）	8	311,850	0	0	0	完全一致

表 18 各ブロックの進捗状況と要約

ブロック	状況	要約
wakuc（外枠）	compare 上、主要出力は整合	前提データはおおむね一致
emp_kyufu（給付）	hou2 は strict 0 到達例あり；shusg 等は許容付き pass の記録	給付推計の核心はかなり追いついた
nat（国年）	主要 mapping は pass；ROREI_SHINKI 等は FP 累積で per_file_tolerance	長期ループ由来の微差を許容運用
bas（基礎）	拠出金・2023 年付近は近接；kekka 系で 604 セル不一致の記録	golden 更新または許容の整理が残る
emp_shushi（収支）	cutb 完全一致・cuta は exclude；01shushi 長期はマクロで数%乖離（§5.2）	compare pass と長期 % は別指標
bunpu（分布）	データ不足時は空出力方針	標準シナリオでは優先度低

表 19 パリティ検証における実装漏れの典型例と調査イメージ

種類	例	効果のイメージ
計算順序	fhantei ループの向き（C は逆順）	数%級の差が千分の一程度へ
参照ずれ	NOUFU インデックスの −1 ずれ（bas）	拠出金の大幅欠落
参照ずれ	yuizor の rs 読込行ずれ	ゼロ → 正しい係数へ
条件式	cond_special が空	同上
初期化欠落	zero_sepsd の配列	数倍の過大評価
初期化欠落	dtst のゼロ化	下流の過大
ループ範囲	siml の i ループ	特定指数の欠落
ブロック漏れ	saitezojuk 第 2・3 ブロック	加給経路の欠落

表 20 マクロ指標（厚生年金収支等）の平均・最大乖離率

系列	平均乖離率	最大乖離率	最大乖離年
収入計	1.98%	6.48%	2120
支出計	1.49%	2.02%	2110
収支差	7.06%	14.63%	2120
積立金	4.50%	11.28%	2120
総報酬・保険料	≈0%	≈0%	—

表 21 政策ブロック別ミクロ検証（strict/tol 齟齬率と判定）

ブロック	従来 exe	Python	比較セル数	strict 齟齬セル数	strict 齟齬率	tol 齟齬率	compare 判定
外枠	wakuc.exe	wakuc	4,616,343	1	≈0%	≈0%	実質完全一致 (FP 域)
厚生・給付費	usys20.exe	emp_kyufu	22,811,138	0	0%	0%	完全一致
国民年金	nat.exe	nat	2,182,434	1,817	0.083%	0%	許容内一致（微差・一部 rel≈1 セル）
基礎年金	bas.exe	bas	5,521	1,186	21.5%	21.5%	要整理（kekka 2 本・前提差）
厚生・収支	asys20.exe	emp_shushi	18,905	0	0%	0%	compare 上は完全一致（cuta 等は exclude）
分布	bunpu.exe	bunpu	311,850	0	0%	0%	完全一致

表 22 基礎年金モジュール（bas）の主要指標における R 版と Python 版の比較

指標	Golden	Python	差
2023 年度 基礎年金拠出金	約 3.0×10^{12} 円	約 3.0×10^{12} 円	実質一致
2023 年度 年度末積立金	約 1.366×10^{13} 円	約 1.365×10^{13} 円	実質一致
TUMATUMI ファイル	777 セル	不一致 0 セル	完全一致

表 23 ブロック別の最大乖離率一覧

政策ブロック（§2.5.1）	検証	対象（代表）	最大乖離率	備考
外枠	ミクロ	waku1000-12.csv 他 100 ファイル	≤0.01%	strict 齟齬 1 セル（FP 域）
厚生・給付費	ミクロ	hou2 / shusg / kisor 等 21 ファイル	0	hou2 含め strict・tol とも 0
国民年金	ミクロ	ROREI_SHINKI	≤0.01%	strict 1,519 セル・最大 rel は極小
国民年金	ミクロ	SHOGAI 等（その他 4 ファイル）	100%（1 セル）	zero_vs_nz 1 件・他は ≤2% 帯
基礎年金	ミクロ	kekka1000-*（2 ファイル）	99.46%	終了年度・cuta 前提差（§4.3）
厚生・収支	ミクロ	cutr / 90nenbe 等（compare 対象 12 ファイル）	0	01shushi 収支見通し本体は §2.1.5 の盲点
厚生・収支	マクロ	収入計（01shushi_sum）	6.48%	2120 年・§5.2.1 表
厚生・収支	マクロ	収支差（01shushi_sum）	14.63%	同上
厚生・収支	マクロ	積立金（01shushi_sum）	11.28%	同上
分布	ミクロ	分布出力 8 ファイル	0	—

表 24 収入計（マクロ指標）のパリティ改善推移

段階（目安）	収入計 平均	収入計 最大（2120 ごろ）	主な要因
悪化世代	約 839%	約 1912%	waku/shus 世代不整合・多 pseid 連続実行
中間	約 14–34%	約 35–94%	golden waku・制度別 emp_kyufu 整合
記録最良（2026-05-19 頃）	3.25%	10.14%	基本層整合後の emp_shushi 再実行
今回（最新プロット・2026-05-22 再生成）	1.98%	6.48%	フルパイプライン後の plot_emp_shushi_parity.py 出力
直前世代（参考）	4.45%	13.07%	同一指標・別実行タイミング

表 25 パリティ検証結果の報告における状況区分

区分	内容
達成	報酬・保険料は実質一致；収入計は 桁違いの悪化世代からは大幅改善
残課題	収支差・積立金は終期で 10〜15% 帯；01shushi の compare 盲点は §2.1.5 のとおり
注意	emp_shushi_ez_arev_shushi の strict pass と、本マクロ指標は別

表 26 付録：リポジトリ内の参照ドキュメント一覧

文書	用途
docs/PARITY_OVERVIEW_rev2024.md	パリティ進捗の一枚サマリ・ミクロ／マクロの留保・01shushi compare 盲点
docs/PARITY_DEBUG_GUIDE.md	差分パターン・実装漏れ教訓・FP 許容
docs/PARITY_QUANTIFICATION.md	compare 結果の定量一覧
docs/CURRENT_ISSUES.md	TODO・既知差分・運用手順
docs/MODULE_PARITY.md	モジュール別詳細・長期収支 §C
docs/HANDOFF_NEXT_SESSION.md	直近セッションの乖離構造・残課題（2026-05-23）
docs/COMPARE_DOMAINS.md	compare ターゲット・golden 更新手順
docs/README_TEAM_WORKFLOW.md	チーム向け実行フロー
docs/emp_shushi_parity_stats.csv	長期収支の数値サマリ（図の元データ）

図 1 集計変数の時系列的推移（水準）

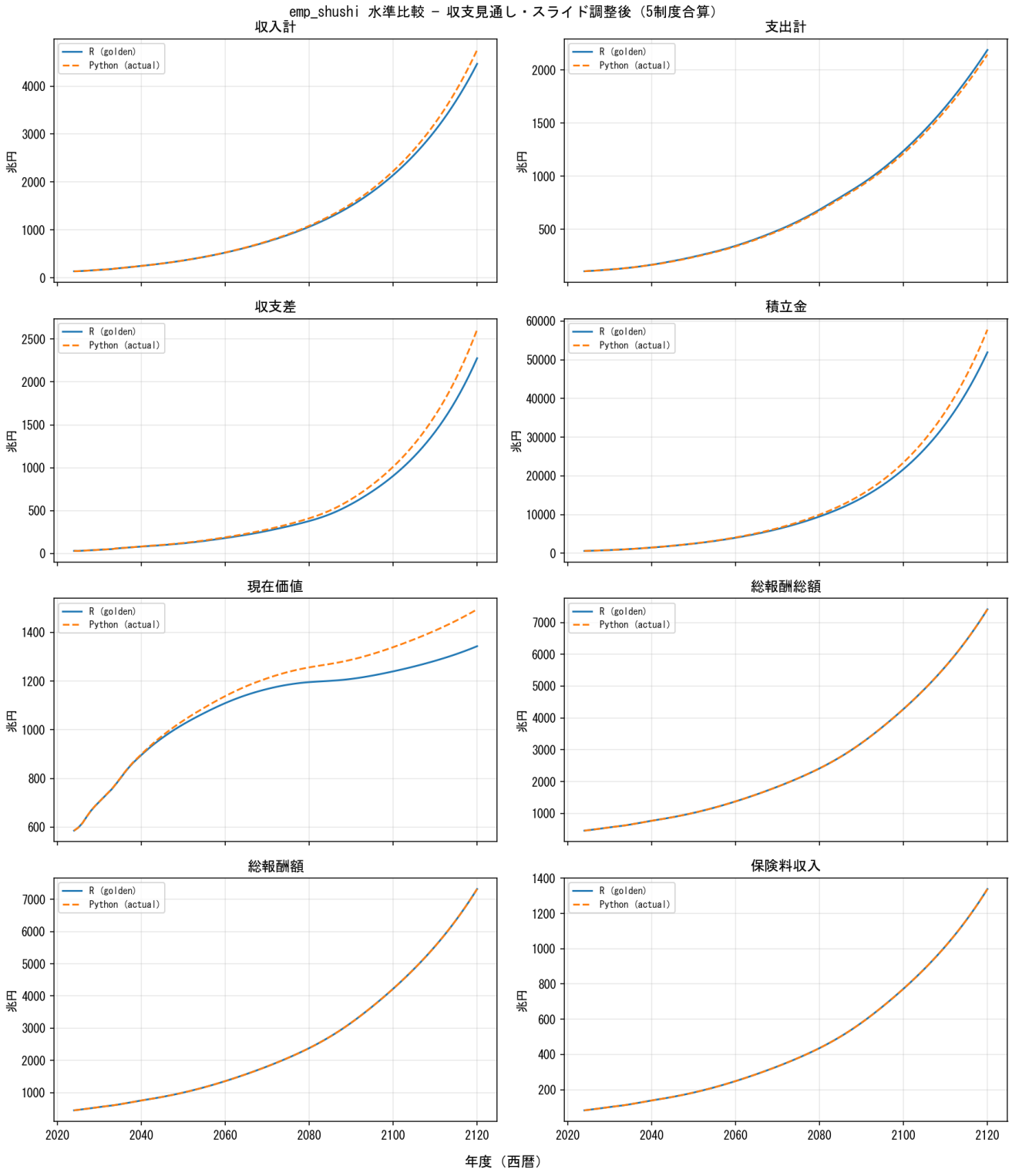
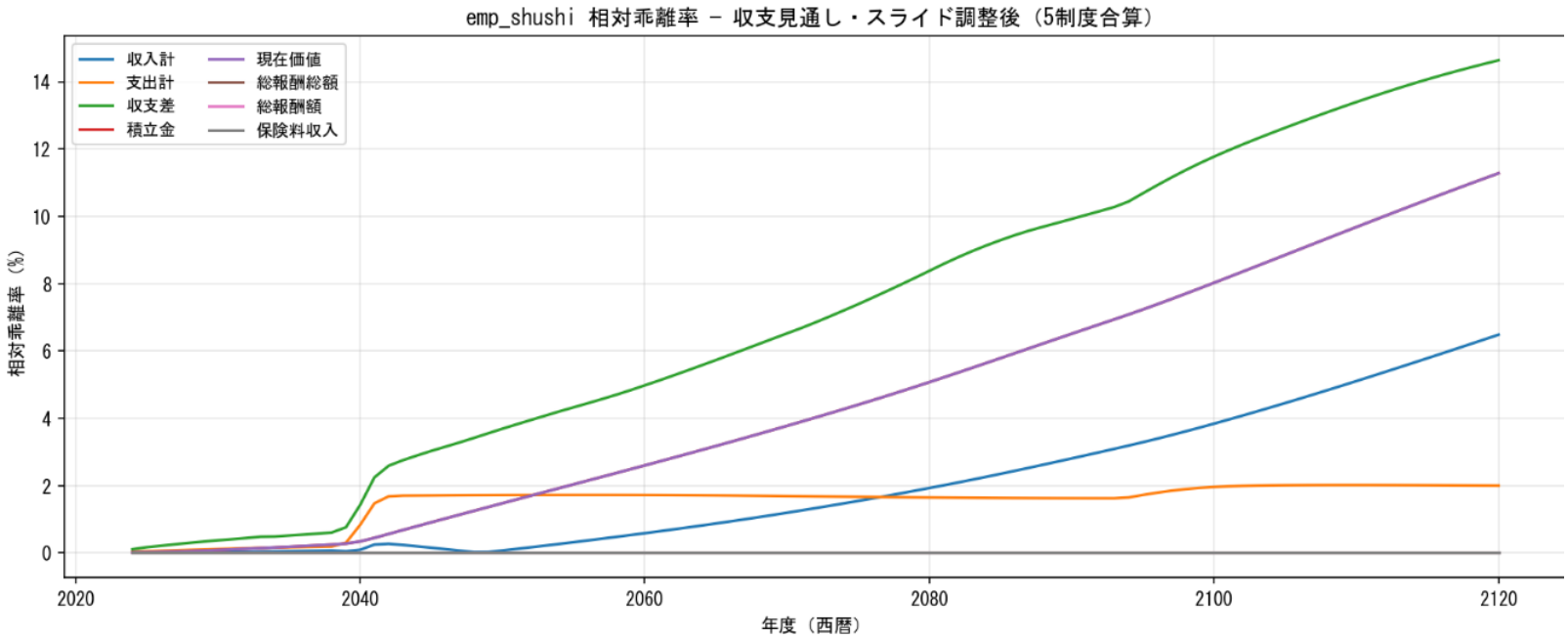


図 2 集計変数の時系列的推移（乖離率）



参考文献

Bagari, A., 2026. AST-based static verification for legacy to python code translation. Journal of Computational Modernization.

Cazzola, M., Favalli, L., 2024. Incremental refactoring of legacy systems: A profiling-based approach. Software Engineering and Modernization.

Challa, S., others, 2025. Semantic equivalence in cross-language model migration. International Journal of Simulation Modelling.

Kukreja, V., others, 2025. Performance optimization of python-based scientific models using numba and cython. Computational Economics.

Nguyen, V., Marché, C., 2011. Formal verification of floating-point c programs using frama-c and Why3. Formal Methods in System Design.

Okamoto, A., 2013. Welfare analysis of pension reforms in an ageing japan. The Japanese Economic Review.

Orvalho, T., Kwiatkowska, M., 2025. Bounded model checking for floating-point precision in migration scenarios. Journal of Software Verification.

Pietrini, R., Paolanti, M., Frontoni, E., 2024. Bridging eras: Transforming fortran legacies into python with the power of large language models, in: ICMI 2024. https://doi.org/10.1109/icmi60790.2024.10586058

Shah, D., 2024. Pensions under pressure: Unraveling the crisis and debating reform in usa's aging society. International Journal for Research in Applied Science and Engineering Technology. https://doi.org/10.22214/ijraset.2024.58447

Suleiman, S., Andongwisye, J., Sinkwembe, E.E., Lundengård, K., 2024. Application of time-varying mortality rate model in pension system: A case of tanzania. Tanzania Journal of Science 50, 244–252. https://doi.org/10.4314/tjs.v50i2.6

Sun, W., 2024. Based on the analysis of existing economic policies, compares the response of economic policies to population ageing between the UK and japan. Deleted Journal 9, 70–76. https://doi.org/10.54254/2977-5701/9/2024074

Zarudnyi, O., Koval, R., 2024. Application of probabilistic and stochastic models and data mining for forecasting the contingent of old age pension recipients in the context of systemic uncertainty. Eastern-European Journal of Enterprise Technologies. https://doi.org/10.15587/2706-5448.2024.313960

厚生労働省, 2024a. 財政検証結果レポート 2024. 厚生労働省.

厚生労働省, 2024b. 財政検証結果レポート 2024 別冊. 厚生労働省.

山本克也., 2010. 公的年金の支給開始年齢引き上げに関する財政シミュレーション. 社会保障研究.

橘木俊詔., 岡本章., 川出真清., 畑農鋭矢., 宮里尚三., 島俊彦., 石原章史., 2006. 社会保障制度における望ましい財源調達手段 (Discussion Paper Series No. 06-J-057). RIETI.

蓮見亮., 中田大悟., 2009. 世代重複モデルを用いた年金改革の評価. 経済分析.

鈴木亘., 湯田道生., 川崎一泰., 2003. 人口予測の不確実性と年金財政: モンテカルロシミュレーション. 季刊社会保障研究.

髙橋済., 豊髙裕夢., 佐川明那., 2023. 年金財政検証の推計プログラムの利用方法とその応用 (財務総研スタッフ・レポート No. No.23-SR-02). 財務総合政策研究所.