AI for Economic Research: Dynamic Models, Language, and Agents
Lecture 9.3: The Homotopy Workflow and Tool Choice

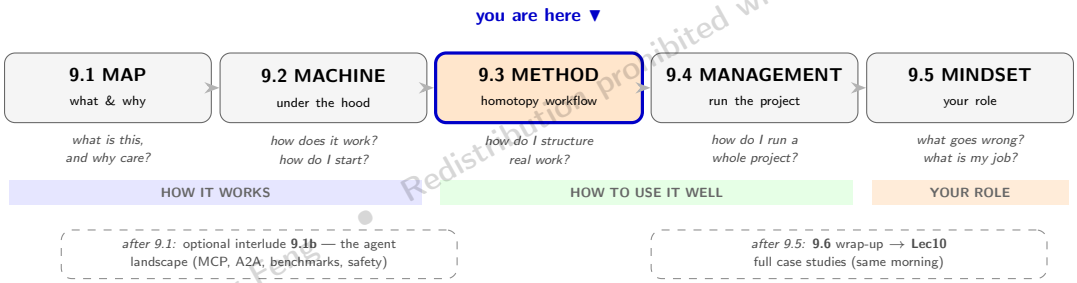
Zhigang Feng

2026-07-06 07:06:03

© Zhigang Feng
Redistribution prohibited without permission

The Map: Act 3 — The Method

This deck answers the third question: how do I structure real work with the machine I just learned to drive?



Redistribution prohibited without permission
© Zhigang Feng

Route: the homotopy idea → the six-step pipeline, step by step (with HA-Ramsey's actual opening prompts) → how to grade a plan before approving it → the runnable OLG demo → which tool for which step.

The Homotopy Workflow

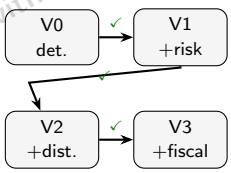
Build from simple to complex, validate, then extend

© Zhigang Feng • Redistribution prohibited without permission

The Homotopy Idea: Solve the Easy Version First

In workflow design, homotopy means solving an easier version first and then extending it toward the full research task.

- Start from a version with clear intuition and known checks
- Use that validated baseline as the starting point for the next
- Attribute new bugs to the new margin you just added
- Never ask the agent to solve the full complicated problem in one jump

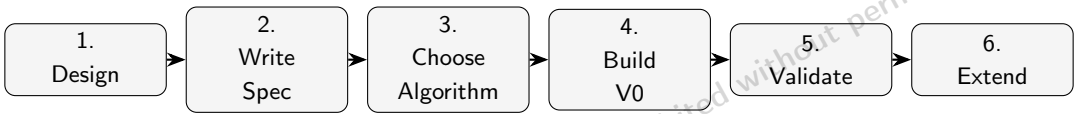


Each step small; each step validated; if a step fails, return to the last validated version.

© Lingang Feng Redistribution prohibited without permission

Why the name: homotopy continuation starts from a problem with a known solution and *continuously deforms* it toward the target (Judd 1998, Ch. 5–7). V0 → V1 → V2 are gates along that deformation — the same discipline, scaled up, produced HA-Ramsey (Story 1; case in Lec10 T8).

Homotopy in Practice: Build from Simple to Complex



This replaces the fantasy prompt "solve my project."

- Start with a problem you can state and benchmark
- Give the agent bounded tasks with visible acceptance criteria
- Add one new ingredient at a time

The next frames walk the six steps in order — with the HA-Ramsey case showing what each step looks like at research scale.

© Zhigang Feng Redistribution prohibited without permission

Step 1 — Design: State the Goal Like an Economist

Before any code, write down the task in economist language.

- Research question in one sentence
- Objective and constraints, or the relevant source rules
- Equilibrium definition, estimand, or review standard if applicable
- Parameters, sample, source universe, and economic motivation
- Outputs you will treat as evidence

Aiyagari anchor:

$$\max \ \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t.} \quad c_t + a_{t+1} = (1+r)a_t + wz_t, \ a_{t+1} \geq \underline{a}$$

© Zhigang Feng • Redistribution prohibited without permission

Goal template (T2's step 0, made precise): *"Compute X with method Y to answer Z; success looks like W."* If you cannot fill in W, you are not ready to open the terminal.

Step 2 — The Spec Is the Contract

The specification file is the contract between you and the agent.

A good specification file states four things clearly:

1. assumptions and domain rules
2. data or source restrictions
3. procedure and validation target
4. required outputs and where to save them

Examples: CRRA utility and convergence tolerance for Aiyagari; weights and subgroup cuts for MEPS; quote-level evidence rules for paper review; official/institutional-source rules and provenance fields for fellows data.

Best practice: ask the agent to restate the specification in its own words before it writes code. That catches misunderstandings when they are still cheap.

© Zhigang Feng Redistribution prohibited without permission

How HA-Ramsey Opened: Restate, Audit, Refine

The restate-first practice is not hypothetical — these three prompts opened the HA-Ramsey collaboration.

```
1. "Here are the RA paper (TeX) and the RA repository.
    Restate the model and the algorithm in your own words.
    List every equation and where it is implemented."
2. "Audit the code against the document. Report every place
    where the code and the TeX disagree: scoring, bounds,
    sampling, penalties. Classify each as a correctness risk
    or a documentation gap."
3. "Propose a refined RA document + code pair in which the
    two are consistent. Do not add any new economics yet."
```

© Zhiang Fang • Redistribution prohibited without permission

- Prompt 1 tests comprehension; prompt 2 tests the *seed*; prompt 3 fixes the foundation while freezing the economics
- "The audit is the cheapest insurance in the whole project" — extending an inconsistent seed propagates the inconsistency

Source: HA-Ramsey case-study package (Lec10 T8).

Grade the Plan Before Any Code: Five Criteria

T2 said: start in plan mode, review, approve. Approve against what? These five criteria.

1. **Restates your goal correctly** — in its own words, not your words echoed back
2. **Bounded steps** — each step has a named deliverable small enough to review in minutes
3. **Acceptance criterion per step** — a benchmark, test, or check named *in advance*
4. **Stop and rollback points** — what happens when a step fails; which validated version you return to
5. **Smallest testable object first** — the plan begins with V0, not with the full target

A plan without acceptance criteria is a wish. Send it back — a plan revision costs one message; a wrong build costs a day.

Fellows instance: one batch of 50 = a bounded step; "every filled field carries a source URL" = its acceptance criterion. HA-Ramsey instance: the audit prompt is criterion 3 applied to the seed itself.

© Zhigang Peng · Redistribution prohibited without permission

Step 3 — Algorithm and Pseudocode Catch Design Errors

Pseudocode or a phase plan is where you catch design mistakes before implementation.

```
Phase A: ingest or define the object
Phase B: run the chosen procedure
Phase C: save diagnostics and checks
Phase D: report only what passed validation
```

Concrete procedures: VFI or EGM for Aiyagari, a 3-prompt pipeline for MEPS, section decomposition for paper review, and batched metadata collection for fellows data.

The point is not elegance. The point is that you and the agent agree on the procedure before code details begin to hide conceptual errors.

© Zhengyang Jiang Redistribution prohibited without permission

Step 4 — V0: The Simplest Version With a Benchmark You Understand

V0 is the simplest version with a benchmark you understand — ideally a closed form.

Optimal growth V0: full depreciation. With log utility, Cobb–Douglas output $y_t = z_t k_t^\alpha$, and $\delta = 1$, the model has an exact closed-form solution:

$$k_{t+1} = \alpha\beta\, z_t k_t^\alpha, \qquad c_t = (1-\alpha\beta)\, z_t k_t^\alpha$$

- Your V0 solver must reproduce this policy to numerical precision — an *absolute* benchmark, not a plausibility check
- Then relax $\delta < 1$: the closed form disappears, but the validated V0 has already certified the plumbing — grids, interpolation, Euler equation, simulation

Discipline: do not ask the agent to add partial depreciation, risk, or equilibrium search before V0 matches the closed form.

© Zhigang Feng • Redistribution prohibited without permission

V0 at Research Scale: What Makes a Good Seed

At research scale, V0 has a name: the seed. HA-Ramsey grew from a published RA solver because that seed had four properties.

1. **It works** — reproduces its own paper's numbers, not just "runs without error"
2. **It is documented** — a TeX model note in the *same notation* as the code
3. **It is modular** — config-driven, so every later experiment is a config diff, not a code fork
4. **It points somewhere** — the RA model is the degenerate no-risk case of the HA target; the deformation path exists by construction

A good seed is like a good prototype for manufacturing a new product: it already works end-to-end at small scale, so scaling up is engineering and refinement — not a fresh invention on the factory floor.

Same object, different scales: the classroom V0 and the research seed play the same role. The OLG demo's `v0` (later this deck) is your training-wheels seed.

© Zhigang Feng • Redistribution prohibited without permission

What "V0 First" Means Across Four Tasks

Task	V0	First validation target
Aiyagari model	deterministic household problem	recover simple savings logic and boundary behavior
MEPS pipeline	3-year pilot panel	weighted rates line up with an external benchmark
Paper review skill	one section or theorem block	quotes are real and comments are specific
Fellows dataset	seed roster only	source rules and provenance fields are actually recorded

© Zhiheng Feng — Redistribution prohibited without permission

Why economists should care: V0 keeps the first error interpretable. You want the first failure to be obvious, not hidden inside a giant workflow.

Steps 5–6 — Validate, Then Extend One Margin at a Time

Version	New ingredient	Required benchmark
V0	Deterministic household	analytical cases
V1	One new model, data, or review margin	setting the new margin to zero recovers V0
V2	Saved diagnostics and state tracking	mass sums, benchmarks, quote checks, or source logs behave sensibly
V3	Full target workflow	final artifact survives validation and interpretation

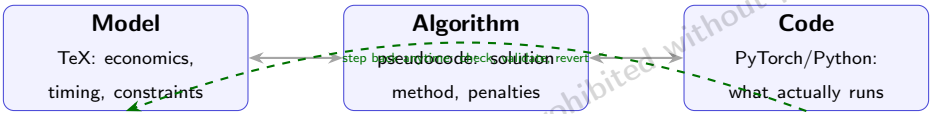
© Zhigang Feng • Redistribution prohibited without permission

Why this works with agents: when a new version breaks, both you and the agent know where to look.

Hold the gate idea: in T4 it returns as a management principle — a quality gate on every intermediate good, not just on code versions.

Three Layers Grow Together: Model, Algorithm, Code

A research project is not one artifact but three, and the homotopy steps move all three in lockstep.



© Arrame Fang · Redistribution prohibited without permission

- **Forward:** you grow from the current version toward the final version gradually — a new model element forces a new algorithm forces new code, one margin at a time
- **Backward:** every earlier version is preserved and validated, so you can always step back to a simpler one — to check and validate a result, to isolate where something broke, or to revert to the last working version
- **The agent's job here:** keep TeX, pseudocode, and code describing *the same economy* at every version — drift between layers is where silent errors live

Source: HA-Ramsey case-study package (Lec10 T8).

Why This Beats One-Shot Generation

Four reasons the homotopy workflow is especially effective with agents:

1. **Agents excel at well-specified, bounded tasks.**
   "Solve my model" fails. "Implement V0 per this pseudocode and validate against these 3 benchmarks" succeeds.
2. **Starting simple gives both you and the agent a working codebase.**
   V1 inherits validated V0 code. The agent builds on tested functions, not a blank file.
3. **Bug attribution is immediate.**
   If V2 breaks, the bug is in V2's new lines, not the validated inherited ones.
4. **Context builds naturally.**
   By V3, the agent has seen the spec, pseudocode, three validated versions, and every correction. *Deep context* — not a 10-page prompt.

Discipline: resist the urge to skip ahead. A working V0 takes 30 minutes; debugging a broken V2 because V0 was never validated takes a day. This is also how good RAs are trained: simple task, verify, widen scope.

© Zhigang Feng Redistribution prohibited without permission

OLG Five-Step Demo

A concrete homotopy workflow for computational macro

© Zhigang Feng • Redistribution prohibited without permission

Demo: Folder Structure Is the Method

Folder or file	Teaching role
`stages/`	five durable stages: model, equilibrium, algorithm, pseudocode, implementation/validation
`prompts/v*_to_v*.md`	one bounded terminal-agent session per version transition
`versions/v0-v5`	validated working code preserved at every step
`tests/`	one smoke test per version; failures identify the new margin
`build/*.json`	reproducibility reports from terminal runs

Demo path: `demos/M4_olg_5step_demo/` (local copy in this lecture; mirror of MiniCourse_8hr `M4_olg_5step_demo` minus `.venv/build/`).

Note the pattern: the HA-Ramsey `Case_Study/` package (Lec10 T8) is this same folder discipline at paper scale — seed, versions, notes, audits, all preserved.

© Zhigang Feng • Redistribution prohibited without permission

Version Ladder: One Economic Margin at a Time

Ver.	Adds	Validation gate
V0	3 cohorts, 1 asset, 2-state TFP, one NN policy	deterministic steady-state collapse; procyclical capital; RMS Euler residual below threshold
V1	7 cohorts and lifecycle labor	V0 logic preserved; middle-aged consumption peak appears
V2	4-state Markov TFP	collapse back to two states reproduces V1
V3	bonds, price, clearing, complementarity	zero bond-loss weight reproduces V2; positive late-life holdings
V4	capital adjustment cost	zero adjustment cost reproduces V3; capital path smooths
V5	stabilizing homotopy phases	residuals fall by phase; final Euler error target met

Research lesson: the agent never receives "solve the big model." It receives one validated deformation at a time.

© Zhigang Feng
Redistribution prohibited without permission

Demo Launch and Classroom Contract

```
cd Lec09_Agentic_AI/demos/M4_olg_5step_demo
pip install -e .
python3 -m unittest discover -s tests
python3 run_demo.py
jupyter notebook demo.ipynb
```

- **What the demo proves:** homotopy preserves a working baseline while complexity rises.
- **What to inspect:** prompts, version folders, test names, validation messages, and regenerated reports.
- **Failure recovery:** if V3 breaks, return to validated V2 and add only the bond-market-clearing layer.

© Zhigang Feng • Redistribution prohibited without permission

Tool Choice

Match the tool to the pipeline step

© Zhigang Feng
Redistribution prohibited without permission

Tool Choice per Pipeline Step

The six-step pipeline also tells you which tool to open.

Tool type	Pipeline steps	Economic example
Web chat	1–2: design, draft the spec	clarify identification logic, draft a model summary
IDE assistant	single-file edits inside a step	vectorize one function, explain one traceback
Terminal agent	4–6: build, validate, extend	restructure the project, run scripts, generate diagnostics, iterate

© Zhigeng Geng · Redistribution prohibited without permission

Why the terminal agent becomes the workhorse: it operates on the real project and runs the real code — while still leaving you in the approval and validation loop.

For empirical research this is usually the sweet spot: enough autonomy to save time, not so much autonomy that you stop checking the economics.

Bridge: One Ladder vs. the Whole Factory

Act 3 disciplined the growth of one artifact. A real project is a factory: many artifacts, many sessions, several reviewers.

- You now have the *method*: design, spec, algorithm, V0, validate, extend — with a plan graded before any code
- But a paper is model + algorithm + code + data + reviews + write-up, worked across weeks of sessions
- The next question: *"how do I run the whole project?"* — Act 4's answer is a role change: the researcher as **project manager**, and the project objects (CLAUDE.md, skills, agents, rules) as the manager's instruments

*Arc: T1 Framing → T2 Under the Hood → **T3 Homotopy** → T4 Project Org → T5 Mistakes/Context → T6 Wrap.*

© Zhigang Feng • Redistribution prohibited without permission