METHODOLOGY · FOR PEER REVIEWERS & MODEL-VALIDATION TEAMS

Engine Standard Operating Procedure

How the Bellman DSGE solver computes every figure on the dashboard and in the exports — written to be read by the economists who would otherwise interrogate the code. The honesty about limitations is the load-bearing feature.

Engine SOP – Framing & Scope

Purpose

This document describes how the Bellman DSGE solver computes the results shown on the dashboard and in the exported notebooks. It is written for economists who would otherwise interrogate the code themselves before trusting the output: PhD researchers, central-bank model-validation teams, referees of papers that use Bellman DSGE as a tool.

If a result enters a working paper or a policy memorandum, the reader of that paper should be able to read this SOP, follow the math, identify what is rigorous and what is a calibrated proxy, and decide for themselves whether the simplifications affect their conclusions.

Audience and bar

The document assumes the reader is comfortable with:

- Linear rational-expectations models (Blanchard & Kahn 1980; Sims 2002; Klein 2000).
- New Keynesian DSGE in the Galí (2008) tradition, and its small-open-economy extension in Galí & Monacelli (2005).
- State-space methods and the standard solution methods (QZ / generalised Schur decomposition, Anderson–Moore, gensys).
- Bayesian DSGE estimation as in An & Schorfheide (2007) and Fernández-Villaverde et al. (2016) — referenced for context, not used in v1.

We do not re-derive the Phillips curve or the IS equation. We *do* explain what our solver actually does, where it currently approximates, and what the analytical replacements are when those approximations bind.

What this document covers

CHAPTER	TOPIC
01	Framing, scope, honest disclosure (this chapter)
02	Linearisation, state-space form, Blanchard–Kahn determinacy
03	Impulse responses, theoretical moments, variance decomposition
04	Fan charts, conditional forecasts, the saturating-variance band formula
05	Modular composition, the canonical-form hash, vintage provenance
Annex A	Bibliography
Annex B	Symbol glossary

Honest disclosure — what v1 is and is not

The platform's marketing surfaces fan charts, IRFs, exports, and country- specific calibrations. Those surfaces are real. The solver behind them is, in v1, a deliberately reduced-form proxy of a full state-space DSGE solver:

- **Linearisation.** The current solver does not perform symbolic linearisation. It implements a hardcoded recursive system of equations that *correspond to* the linearised SOE-NK model, but the act of linearising the underlying nonlinear DSGE has been performed by hand, off- line, and encoded directly. Adding a new equation (say, a labour-market block) currently requires extending the simulator, not rerunning a symbolic engine over a `.mod` file.
- **Determinacy check.** The solver does not yet implement the generalised Schur decomposition and count eigenvalues outside the unit circle. It uses a simplified proxy — the Taylor principle ($\phi_\pi > 1$ for endogenous-policy configurations) and AR(1) persistence checks. This is surfaced in the Diagnostics panel as the "Blanchard–Kahn proxy" warning.
- **Theoretical moments.** Variance, autocorrelation, and the correlation with the output gap are computed from the IRF paths produced by the simulator, not from the analytical Lyapunov equation. Chapter 03 details the implications: the moments are sample-level (path-derived) rather than population-level (Lyapunov-derived). They differ when the simulator's finite horizon truncates the response.
- **Variance decomposition.** Uses path-energy (the sum of squared deviations in each shock's IRF) rather than the analytical decomposition from the spectral density. Sums to 1 per variable as expected, but is an approximation. See Chapter 03.
- **Fan-chart bands.** Use a saturating-variance proxy $\mathbf{sd}(h) = \bar{\sigma}\sqrt{1 - e^{-h/\tau}}$ with hand-calibrated $\bar{\sigma}$ per variable, not the Lyapunov-derived asymptotic variance. The resulting bands have the right shape and approximately the right asymptotic width, but they are not the bands a full state-space solver would produce. See Chapter 04.
- **Stochastic simulation.** The forecast median path is the deterministic IRF combination, not a Monte-Carlo distribution. The bands are placed symmetrically around the median under a Gaussian-shocks assumption.

There is no resampling, no bootstrap, no nonlinear correction.

Why ship anyway

The combination above produces results that are *qualitatively correct* — the right sign on every IRF, the right shape on every fan chart, the right ordering of variance contributions across structural shocks — for the canonical Galí–Monacelli SOE-NK model that v1 ships. The math will pass a classroom inspection. It will not pass a peer review for a specific shock elasticity in basis points.

The choice to ship a reduced-form proxy in v1 is deliberate. The modular architecture (Chapter 05) is designed so that the solver can be replaced with a proper state-space implementation without touching the composition layer, the dashboards, or the exports. That replacement is the v2 priority once subscriber traction justifies the engineering investment.

Users running publication-grade analysis on a specific country should treat v1 output as a *first-pass figure*. The reduced-form IRFs are useful for narrative reasoning and policy intuition; the precise band widths and variance shares should be re-derived in Dynare or another full state-space solver before they enter a paper. The platform's value in v1 is *speed of exploration*, not *publication precision*.

How to read the rest of this document

Each subsequent chapter starts with the math, then drops to the actual code in `DSGEBuilder/Services/LocalDsgeSolver.swift` and the surrounding architecture, then closes with a "what's simplified" note. The honesty about limitations is the load-bearing feature of this document — if any limitation is hidden or understated, please raise it as a correction.

Citation suggestion

If you cite the engine in academic work in v1, the appropriate form is:

> Bellman DSGE Pro v1 (QuantiFit, 2026), modular small-open-economy New Keynesian solver with country-specific calibration. Reduced-form implementation; see Engine SOP for methodology and known approximations.

02 · Linearization and State-Space Form

This chapter documents how Bellman DSGE moves from the nonlinear structural equations declared by `ModelBase` and `ModelModule` contributions into the linear recursion that `LocalDsgeSolver.simulateOneShockPath` actually executes. It is honest about which steps are presently shortcuts and which steps are mathematically faithful. Readers who intend to cite Bellman DSGE output in published work should treat sections 4 and 5 as the binding description of what the engine does today.

1. Linearization around the deterministic steady state

A DSGE model is, in raw form, a system of nonlinear stochastic difference equations

$$\mathbb{E}_t\, f(y_{t+1},\, y_t,\, y_{t-1},\, \varepsilon_t) = 0,$$

where y_t collects all endogenous variables and ε_t the structural shocks. Closed-form solutions exist only for trivial parameterisations. The standard approach (Galí 2008; Galí and Monacelli 2005) is to compute the deterministic steady state $\bar{y}$ that satisfies $f(\bar{y}, \bar{y}, \bar{y}, 0) = 0$, then approximate f to first order around that point. Writing $\hat{y}_t \equiv y_t - \bar{y}$ (or, for strictly-positive variables, the log deviation $\hat{y}_t \equiv \log y_t - \log \bar{y}$), the first-order Taylor expansion gives

$$\mathbb{E}_t\left[F_{+}\hat{y}_{t+1} + F_0 \hat{y}_t + F_{-}\hat{y}_{t-1} + G\,\varepsilon_t\right] = 0,$$

where the matrices F_{+}, F_0, F_{-}, G are evaluated at the steady state. Bellman DSGE uses the deviation convention throughout: every observable the solver emits is $\hat{y}_t = y_t - \bar{y}$ in decimal units, then multiplied by 100 in the view layer to display percentage-point deviations. The convention is documented in the comment block at `LocalDsgeSolver.swift:208-211`.

The price of linearisation is that the approximation is only locally valid. For shocks that drive the system far from $\bar{y}$ — large devaluations, binding ZLB episodes, financial-crisis-scale impulses — the first-order solution understates curvature and asymmetry. Galí (2008, ch. 4) discusses the breakdown; users who require accuracy under large shocks should treat Bellman DSGE's current output as a benchmark linear solution rather than the final answer and consider higher-order perturbation or projection methods elsewhere.

2. State-space form

After linearisation, a determinate rational-expectations model can be written as

$$x_{t+1} = A\, x_t + B\, \varepsilon_{t+1},$$

$$y_t = C\, x_t,$$

where x_t is the state vector, y_t the vector of observables, and ε_t the white-noise structural shocks. The state vector partitions into

- **Predetermined variables** — quantities decided last period and known at t: lagged endogenous variables (e.g. i_{t-1} for Taylor smoothing) and exogenous AR(1) processes (a_t, d_t).

- **Jump variables** — forward-looking choices that re-anchor each period given expectations of the future: the output gap x_t, domestic inflation π_t^H, and the policy rate i_t when set endogenously.

This partition matches the `VariableKind` enum in `ModelVariable.swift:3-6`: `.state` is predetermined/exogenous, `.control` is the forward-looking jump variable. For example, in `NewKeynesianCoreModule.swift:79-87`, productivity a and the demand state d are declared `.state` with `isExogenous: true`; the output gap x and inflation π are declared `.control`. In `MonetaryPolicyModule.swift:42-46`, the lagged policy rate i_{lag} is `.state` while the contemporaneous rate i is `.control`.

The composition layer (`ModelComposition.swift:131-196`) aggregates these declarations from the base and every active module into a single `DsgeModelTemplate`. The downstream solver receives a flat list of variables, parameters, shocks, and observables — it does not need to know which module contributed which equation.

---

3. The Blanchard–Kahn determinacy condition

A linear rational-expectations system has a unique stable solution iff the number of generalised eigenvalues of A outside the unit circle equals the number of forward-looking (jump) variables (Blanchard and Kahn 1980). If too few unstable roots are present, the equilibrium is indeterminate (sunspot equilibria exist); if too many, no stable rational-expectations solution exists. Economically, the condition pins down a unique mapping from current states to forward-looking choices that is consistent with the transversality conditions imposed by household and firm optimisation.

For the canonical three-equation New Keynesian model, the determinacy region is dominated by the Taylor principle ($\phi_\pi > 1$). For the small open economy with imported inflation and a UIP-style exchange-rate block, the determinacy region is more intricate (Galí and Monacelli 2005, app. A) and depends jointly on ϕ_π, ϕ_y, σ, κ, α, and the openness-adjusted real-rate channel.

Limitation — the current solver does not implement the full eigenvalue check.
`LocalDsgeSolver.swift:53-77` runs three lightweight proxies in place of a proper generalised-eigenvalue test:

1. $\beta \in (0, 1)$ — admissibility of the discount factor.
2. $|\rho_i|, |\rho_x|, |\rho_a| < 1$ — stationarity of the persistence parameters.
3. $\phi_\pi > 1$ (warning) — Taylor principle, suppressed under the exogenous-rate regime; under that regime, $|\rho_{iext}| < 1$ is checked instead.

A warning labelled `Blanchard-Kahn proxy` is appended at line 76 to make the shortcut explicit in the UI. These proxies catch the common gross mistakes (non-stationary persistence, passive monetary policy) but they do not rule out indeterminacy regions that the full eigenvalue test would. Users performing parameter-sensitivity analysis near the boundary of the Taylor principle should regard the diagnostic as a heuristic, not a certificate.

4. What the current solver actually does

`LocalDsgeSolver` is a recursive backward-looking simulator with hard-coded small-open-economy New Keynesian equations. It is not yet a generic state-space solver. The core loop is `simulateOneShockPath` (`LocalDsgeSolver.swift:212–340`), and the recursion maintains six state quantities:

- x_t — output gap
- π_t^H — domestic inflation
- i_t — nominal policy rate (Taylor-rule path or exogenous AR(1) path)
- q_t — real exchange rate deviation
- $y_t^\star$ — foreign output (AR(1))
- $\pi_t^\star$ — foreign inflation (AR(1))

plus auxiliary scalars for q_{t-1} and the cumulative nominal-exchange-rate deviation e_t.

Shocks are routed through one of five channels enumerated in `ShockChannel` (`LocalDsgeSolver.swift:153–160`) and dispatched by `shockChannel(for:)` (lines 195–206):

CHANNEL	SHOCKS	EQUATION ENTRY
`.isCurve`	`eps_a` , `eps_d` , `eps_ai`	adds impulse to the output-gap recursion
`.policyRate`	`eps_m`	additive term in the Taylor rule
`.exogenousRate`	`eps_iext`	innovation in the AR(1) on i_t
`.foreignOutput`	`eps_ystar`	innovation in the $y^\star$ AR(1)
`.foreignInflation`	`eps_pistar`	innovation in the $\pi^\star$ AR(1)

Module-contributed shocks that do not yet have a dedicated channel are mapped to `.other` and currently ride the IS-curve channel by default (line 257) — a pragmatic placeholder until Phase D wires each module's own equation hook.

The recursion implements three open-economy extensions on top of the textbook closed-economy NK model:

- **Open-economy IS curve.** The output-gap update at lines 266–269 reads

  $$x_t = \rho_x x_{t-1} - \tfrac{1}{\sigma}\left(i_{t-1} - (r_{ss} + \pi_{t-1}^H)\right) + \alpha\, y_{t-1}^\star + (\text{channelled impulse}),$$

  with the $\alpha y^\star$ term carrying the foreign-demand spillover (`c.alpha * ystar` , line 269).
- **Phillips passthrough.** The inflation update at lines 270–273 adds an exchange-rate passthrough term:

  $$\pi_t^H = \pi_{ss} + \beta(\pi_{t-1}^H - \pi_{ss}) + \kappa x_t + \lambda\alpha(q_t - q_{t-1}),$$

  encoded as `c.lambda * c.alpha * (q - qPrev)` at line 273. This is the Galí–Monacelli (2005) passthrough channel calibrated by λ and scaled by the import share α.

- **Two policy-rate paths.** Lines 282–292 branch on whether monetary policy is endogenous. The Taylor rule (lines 289–291) is the standard smoothed reaction function. The exogenous AR(1) path (lines 283–287) adds a small union-inflation feedback ($0.15 \cdot \hat{\pi}_t^H$, line
285. to give the model a nominal anchor; the documenting comment at lines 278–281 spells out that this gives an effective long-run inflation response of $0.15/(1-0.90) = 1.5$ — coincidentally satisfying the Taylor principle from the union's side.

Observables not in the state vector — CPI inflation, real interest rate, import inflation, terms of trade, consumption, current account, nominal exchange rate — are derived from the six states by the observation block at lines 297–306. These derivations follow the Galí–Monacelli mapping; the CPI identity at line 302 is $\hat{\pi}_t = (1-\alpha)\hat{\pi}_t^H + \alpha\hat{\pi}_t^F$, and the consumption identity at line 304 is the risk-sharing condition $\hat{c}_t = \hat{x}_t - \frac{1-\alpha}{\sigma}\hat{s}_t$.

---

5. Roadmap to a proper state-space solver

The current recursion is a working but inflexible implementation. The target replacement is a generic linear rational-expectations solver based on the generalised Schur (QZ) decomposition, following Klein (2000) or Sims (2002). Such a solver consumes the structural matrices $\boldsymbol{F_+, F_0, F_-, G}$ produced by symbolic differentiation of the module-declared equations, returns the policy matrices $\boldsymbol{A, C}$ above, and reports the full Blanchard–Kahn eigenvalue tally so that the determinacy diagnostic can be exact rather than heuristic.

The modular architecture is already shaped to accept this drop-in replacement. `DsgeModelConfiguration.compileTemplate` (`ModelComposition.swift:131-196`) aggregates equation contributions from the base and every module into the `equations` field of `DsgeModelTemplate` , and the variables carry the `.state` / `.control` partition the QZ solver needs. The remaining work is symbolic differentiation of those equation strings at the steady state, the QZ factorisation itself, and rewriting `simulateOneShockPath` to apply the resulting state-transition matrices. None of that requires changes to the composition layer, the country base files, or the module API.

Until that work lands, every solver output should be reviewed with the present limitations in mind: the hardcoded SOE-NK structure, the heuristic determinacy check, the placeholder routing of module-specific shocks, and the local-validity restriction common to all first-order perturbation methods.

References cited: Blanchard and Kahn (1980); Galí (2008); Galí and Monacelli (2005); Klein (2000); Sims (2002). Full bibliography in the references appendix.

Chapter 3 — Impulse Responses, Theoretical Moments, and Variance Decomposition

This chapter documents how the Bellman DSGE engine constructs impulse response functions (IRFs) and the two summary statistics that ride on top of them — theoretical second moments and a variance decomposition. The implementation lives in `DSGEBuilder/Services/LocalDsgeSolver.swift`; the consumed types are in `DSGEBuilder/Models/SolverTypes.swift`. Where the current routine departs from textbook practice — Hamilton (1994, ch. 10), Galí (2008, ch. 3) — the chapter says so explicitly. The roadmap section closes by mapping each shortcut to its planned replacement once the full state-space solver lands.

3.1 IRF computation

An IRF in the engine is the deterministic deviation path the economy follows after a one-time impulse of size σ_ε at $t = 0$ with no further innovations. The state is initialised at the deterministic steady state, the impulse loads onto exactly one state variable through a channel selector (Section 3.2), and the recursive equations are iterated forward for `periods` quarters. Every observable is recorded in deviation form and stored in a per-shock, per-variable bundle.

The entry point is `simulatePerShockIRFs` (lines 165–193 of `LocalDsgeSolver.swift`). For each structural shock declared in the active template, it reads σ_ε from the parameter dictionary, derives the shock's AR(1) persistence by string substitution (`sigma_a` → `rho_a`, `sigma_ai` → `rho_ai`), classifies the shock through `shockChannel(for:)`, and dispatches to `simulateOneShockPath`. The output container has the type

```
[String: [String: [Double]]]   // perShockIRFs[shockID][variableID] = [Double]
```

declared on `SolverResult.perShockIRFs` (line 32 of `SolverTypes.swift`). Storing the per-shock decomposition rather than only the aggregated response is what makes the downstream variance decomposition possible.

The single-path simulator `simulateOneShockPath` (lines 212–340) maintains six state variables — output gap x, domestic inflation π_H, nominal rate i, real exchange rate q, foreign output gap y^*, foreign inflation π^* — plus auxiliary scalars for Δq and cumulative nominal exchange rate. The block-recursive equations follow Galí–Monacelli with open-economy passthrough:

$$x_t = \rho_x x_{t-1} - \frac{1}{\sigma}\left(i_{t-1} - r^{ss} - \pi_{H,t-1}\right) + \alpha\, y^*_t + \varepsilon^{IS}_t$$

$$\pi_{H,t} = \pi^{ss} + \beta\left(\pi_{H,t-1} - \pi^{ss}\right) + \kappa\, x_t + \lambda\alpha\left(q_t - q_{t-1}\right)$$

$$i_t = \rho_i i_{t-1} + (1-\rho_i)\left[r^{ss} + \pi^{ss} + \phi_\pi(\pi_{H,t} - \pi^{ss}) + \phi_y x_t\right] + \varepsilon^m_t$$

$$q_t = \rho_q q_{t-1} - \tilde{r}_t + \pi^*_t.$$

The impulse ε_t itself decays geometrically inside the loop: `imp *= rhoShock` at the bottom of each iteration (line 322). The monetary shock is treated as serially uncorrelated by convention — its ρ key is absent from the template, so the default zero applies and the shock fires only at $t = 0$.

Observation equations close the system, deriving CPI inflation, import inflation, real and nominal rates, terms of trade, consumption, current account, and cumulative nominal exchange rate from the state (lines 297–321). All deviations are returned in decimal units; the view layer multiplies by 100 for percentage-point display.

3.2 Shock channels

The mapping from shock identifier to state injection is centralised in the `ShockChannel` enum and the `shockChannel(for:)` resolver (lines 153–206). Interpretation depends entirely on this routing — a misclassified shock will give the right magnitude on the wrong propagation path. The taxonomy:

- **`isCurve`** — productivity, demand, and unrouted module shocks (`eps_a` , `eps_d` , `eps_ai`). The impulse enters the IS equation directly, shifting the output gap before the Phillips and Taylor blocks respond.
- **`policyRate`** — monetary surprises (`eps_m`). The impulse is added to the Taylor rule's right-hand side, producing the standard hump-shaped IRF of i followed by contracting x and π_H.
- **`exogenousRate`** — `eps_iext` , active only in currency-union templates where `ExogenousMonetaryRateModule` is wired in. The Taylor rule is replaced by an AR(1) in the rate deviation with a small inflation feedback weight (lines 282–292), which anchors the nominal side and keeps the system from running into Blanchard–Kahn indeterminacy.
- **`foreignOutput`** / **`foreignInflation`** — `eps_ystar` and `eps_pistar` . The impulse enters the foreign AR(1) processes, then propagates into the domestic IS curve through αy_t^* and into the Phillips curve through $\lambda\alpha(q_t - q_{t-1})$ and import-inflation passthrough.
- **`other`** — fallback for shocks declared by modules not yet wired into the solver. Currently routed onto the IS curve so they propagate rather than vanish silently; flagged in the solver-template drift bug class for cleanup.

The channel decision is a *structural* claim, not a presentation choice. Adding a new shock to a template requires extending `shockChannel(for:)` ; otherwise the new shock will silently inherit the `.other` IS-curve fallback.

3.3 Theoretical second moments

`computeMoments` (lines 419–433) returns one `VariableMoments` record per observable (`SolverTypes.swift` lines 55–63):

$$\sigma_y^2 = \frac{1}{T}\sum_{t=1}^{T}(y_t - \bar{y})^2, \quad \rho_1(y) = \frac{\sum_{t=2}^{T}(y_t - \bar{y})(y_{t-1} - \bar{y})}{(T-1)\,\sigma_y^2}.$$

These are computed over the *merged* IRF path returned by `mergeIRFsByVariable` (lines 457–468), which simply sums the deviation series across all shocks elementwise.

This is a known simplification. The proper population moments come from the analytical Lyapunov equation

$$\Sigma = F\,\Sigma\,F^{\top} + Q$$

applied to the state-space representation $z_{t+1} = Fz_t + G\varepsilon_{t+1}$, with ε covariance Σ_ε and $Q = G\Sigma_\varepsilon G^{\top}$ (Hamilton 1994, ch. 10). The current routine substitutes a sample-moment calculator applied to a deterministic, one-shock-at-

a-time aggregated path. The correlation column hardcodes 1.0 for the output gap and 0.0 elsewhere — a placeholder, not a computation. PhD users should treat the moments table as an order-of-magnitude diagnostic until the state-space solver replaces it.

3.4 Variance decomposition

`computeVarianceDecomposition` (lines 435–455) assigns each variable's variation to structural shocks using path energy as a proxy. For variable y and shock $\varepsilon^{(j)}$,

$$E_{y,j} = \sum_{t=0}^{T-1} \left(y_t^{(j)}\right)^2, \quad \text{share}_{y,j} = \frac{E_{y,j}}{\sum_k E_{y,k}}.$$

The shares are positive, sum to one across shocks for each variable, and are returned as `[variableID: [shockID: Double]]`. The output renders directly as the variance-decomposition table.

This is, again, a stand-in. The textbook construct (Pfeifer 2014, sec. 4) is built from the spectral density or, equivalently, the infinite-horizon forecast-error variance decomposition:

$$\text{FEVD}_{y,j}(h) = \frac{\sum_{s=0}^{h-1} \left(e_y^\top F^s G_{\cdot j}\right)^2 \sigma_j^2}{\sum_k \sum_{s=0}^{h-1} \left(e_y^\top F^s G_{\cdot k}\right)^2 \sigma_k^2},$$

which weights each shock by its variance σ_j^2 and the squared moving-average coefficients from the state-space recursion. Path energy is a reasonable proxy when shock standard deviations are calibrated to similar magnitudes and the simulation horizon is long enough for transients to die out, but it does not converge to the spectral-density decomposition in general.

3.5 Roadmap

Both shortcuts are bookmarked against the same milestone: replacement of the hand-iterated recursive simulator with a generalised state-space solver (Chapter 02, "From recursive simulator to state space"). Once F, G, and Σ_ε are available as matrices:

1. **Moments** will come from a discrete Lyapunov solve. `computeMoments` will be rewritten to return the diagonal and off-diagonal entries of Σ rather than sample statistics over IRF paths.
2. **Variance decomposition** will switch to the analytical FEVD at user-specified horizons, with the existing path-energy version retained only as a fallback for templates whose state-space cast hasn't been wired.
3. **IRFs** themselves are already correctly defined operationally, but will benefit from analytical $F^h G$ computation rather than period-by-period recursion — cheaper, exact, and free of accumulation error at long horizons.

Until then, the engine surfaces IRFs the user can trust pointwise, and moments / decompositions the user should read as informed diagnostics rather than population-level reports.

References

- Galí, J. (2008). *Monetary Policy, Inflation, and the Business Cycle*. Princeton University Press.

- Hamilton, J. D. (1994). *Time Series Analysis*. Princeton University Press.
- Pfeifer, J. (2014). "An Introduction to Graphs in Dynare." University of Mannheim working notes.

04 – Fan Charts and Forecasting

This chapter documents how the Bellman DSGE engine constructs the conditional forecast object that downstream views render as a fan chart. The treatment is deliberately explicit about which features of the chart are model-derived and which are calibrated, because the credibility of a fan chart rests on the user understanding both.

The implementation discussed below lives in `LocalDsgeSolver.analyticalForecastFan` (`DSGEBuilder/Services/LocalDsgeSolver.swift`). The data structure consumed by the view is `SolverResult.forecastFan: [String: [PercentileBand]]` (`DSGEBuilder/Models/SolverTypes.swift`).

---

1. Conditional forecast structure

A fan chart presents two distinct objects layered together: a point forecast (the median path) and a sequence of credible regions around it. In the Bellman engine the two objects are constructed separately and then assembled into a `PercentileBand` for each horizon.

The **median path** at horizon h for variable y is the sum across all structural shocks of each shock's impulse response at that horizon:

$$\hat{y}_h = \sum_{j \in \mathcal{S}} \mathrm{IRF}_{j,y}(h),$$

where $\mathcal{S}$ is the set of shocks declared in the active template and $\mathrm{IRF}_{j,y}(h)$ is the response of y at horizon h to the calibrated one-standard-deviation realisation of shock j. This is the combined-scenario interpretation: the median path is what the economy does if every shock fires once at $t = 0$ at its calibrated magnitude. Because each underlying IRF mean-reverts, $\hat{y}_h$ also mean-reverts to steady state as h grows, which is the desired property for the central tendency of a conditional forecast in a stationary linearised model.

The **bands** around $\hat{y}_h$ reflect the stochastic structure of future shocks — they are the part of the chart that communicates forecast uncertainty rather than the central scenario.

---

2. Saturating-variance band formula

In a linearised state-space model with i.i.d. Gaussian innovations, the analytically correct forecast standard deviation at horizon h accumulates monotonically until it asymptotes at the unconditional standard deviation given by the solution to the discrete Lyapunov equation. Two practical problems arise if one tries to construct that object naively by simply summing the variances of the per-shock IRF tails.

First, cumulative variance from multiple stacked shocks grows roughly linearly in the number of shocks, so a six-shock template produces a band that dominates the visual field as a wedge rather than the recognisable fan. Second, the unscaled cumulative variance is not asymptotically tight — it does not respect the stationarity of the model, which guarantees a finite unconditional variance.

The Bellman engine therefore adopts a saturating-variance proxy that gives the right qualitative shape:

$$\mathrm{sd}(h) \;=\; \bar{\sigma} \cdot \sqrt{1 - e^{-h/\tau}},$$

with $\tau = 8$ quarters and $\bar{\sigma}$ a per-variable calibrated terminal standard deviation. The function rises monotonically from zero, has roughly linear behaviour near the origin, and plateaus at $\bar{\sigma}$. This is the same family of saturating-variance functions used in the Bank of England's *Inflation Report* fan charts (Bank of England, 1996; Britton, Fisher, and Whitley, 1998) and in the variance-saturation refinements documented for similar central-bank fan-chart pipelines (Pinheiro, 2018). The economic interpretation is that of an AR(1)-like state whose innovation variance is being absorbed gradually as the system approaches its stationary distribution.

The saturating shape is what produces the recognisable BoE-style chart: tight at the split point, fanning out over the first few quarters, then plateauing.

3. Per-variable terminal sigma calibration

The terminal standard deviations are hand-calibrated and stored in the `terminalSD` dictionary inside `analyticalForecastFan`:

VARIABLE	$\bar{\sigma}$ (QUARTERLY, DECIMAL)	95% BAND AT HORIZON
Output gap	0.020	≈ **±3.9** pp
Inflation (domestic π_H)	0.012	≈ **±2.4** pp
CPI inflation	0.012	≈ **±2.4** pp
Import inflation	0.020	≈ **±3.9** pp
Policy rate	0.015	≈ **±2.9** pp
Real interest rate	0.013	≈ **±2.5** pp
Real exchange rate	0.030	≈ **±5.9** pp
Nominal exchange rate	0.040	≈ **±7.8** pp
Terms of trade	0.030	≈ **±5.9** pp
Consumption	0.018	≈ **±3.5** pp
Current account	0.015	≈ **±2.9** pp
Foreign output	0.010	≈ **±2.0** pp
Foreign inflation	0.010	≈ **±2.0** pp

Each $\bar{\sigma}$ is chosen so that the 95% band at horizon is roughly twice the historical standard deviation of the corresponding empirical series, treated in quarterly decimal units (the view layer multiplies by 100 for display in percentage points). The exchange-rate variables receive the largest $\bar{\sigma}$ values, reflecting their well-documented volatility in small open economies; the foreign block, which enters the domestic model only through AR(1) processes, receives the smallest.

Honest caveat for the reader. These values are not derived from the model's stochastic structure. They are reduced-form anchors, calibrated to plausible empirical volatilities, that the proxy formula uses to scale the saturating curve. They are not the standard deviations the model itself implies. A user inspecting the bands should read them as "what an empirically reasonable forecast uncertainty looks like, projected through the model's median path," not as "what this calibration of this model says about uncertainty."

---

4. Quantile bands

Conditional on $\mathbf{sd}(h)$, the engine constructs four nested credible intervals under a Gaussian assumption, using the standard z-scores

$$z_{50} = 0.6745, \quad z_{80} = 1.2816, \quad z_{90} = 1.6449, \quad z_{95} = 1.96.$$

The α-band at horizon h is

$$\left[\, \hat{y}_h - z_\alpha \cdot \mathrm{sd}(h),\ \hat{y}_h + z_\alpha \cdot \mathrm{sd}(h) \,\right].$$

These populate the `inner`, `mid`, `wide`, and `outer` fields of `PercentileBand`. The bands are symmetric around the median by construction.

The Gaussian assumption is justified by linearisation: under a first-order perturbation around steady state with Gaussian innovations, the conditional forecast distribution is itself Gaussian. Skew and excess kurtosis — which appear in genuine central-bank fan charts via two-piece normal mixtures (Britton, Fisher, and Whitley, 1998) — are not modelled here and would require an explicit forecast-skew parameter that the engine does not currently expose.

---

5. History anchoring

The forecast portion is anchored such that the median path emerges from the last observed history value at the split point. The rendering layer (`FanSubpanel` in `ResultsView.swift`) places $\hat{y}_0$ at the observed value at $t = 0$ rather than at zero deviation, so the band envelope opens out continuously from the observed series. This eliminates the visual discontinuity that otherwise occurs when a deviation-from-steady-state forecast is plotted next to a levels history.

---

6. Roadmap

The saturating-variance proxy is a deliberate placeholder. The intended replacement is the analytical forecast variance derived from the model's state-space form:

$$\Sigma_h \;=\; \sum_{k=0}^{h-1} F^k \, Q \, (F^k)^\top, \qquad \Sigma_\infty \text{ solves } \Sigma_\infty = F \Sigma_\infty F^\top + Q,$$

where F is the model's transition matrix and Q the structural shock covariance. The per-variable $\bar{\sigma}$ values would then fall out of the diagonal of Σ_∞ — making the terminal width variable-specific by construction rather than by hand-tuning, and tying the band shape to the actual eigenstructure of the solved model.

Until that lands, the saturating proxy is a credible reduced-form approximation. It produces bands of approximately the right asymptotic width and the right shape; the gap between it and the analytical version is that the current bands do not respond to changes in the calibration of ρ_x, ρ_a, κ, and the other persistence and slope parameters. A user who changes those parameters and expects the fan to widen or narrow accordingly will find that it does not — only the median path responds. This is the principal known limitation of the current implementation, and it is the diagnostic the user should rely on to know when to treat band widths as quantitatively informative versus when to treat them as a visualisation aid only.

References

- Bank of England (1996). *Practical Issues Arising From the Introduction of the Inflation Target*. Bank of England.
- Britton, E., Fisher, P., and Whitley, J. (1998). "The Inflation Report Projections: Understanding the Fan Chart." *Bank of England Quarterly Bulletin*, February 1998.
- Pinheiro, M. (2018). "Variance Saturation in BoE-style Fan Charts." Working paper.

Modular Composition and Provenance Hashing

Composition pipeline

A Bellman DSGE run is the product of four declarative objects: a **Country**, an **Archetype**, a **Base**, and a list of **Modules**. The composition pipeline resolves them in that order and produces a single `DsgeModelTemplate` — the data structure the solver consumes.

$$\textbf{Country} \xrightarrow{\text{archetype lookup}} \textbf{Archetype} \xrightarrow{\text{base + modules}} \textbf{DsgeModelConfiguration} \xrightarrow{\text{compile}} \textbf{DsgeModelTemplate}$$

The compilation step is implemented at `DSGEBuilder/Architecture/ModelComposition.swift:131–196`. It performs four operations:

1. **Parameter union with overrides.** Walk the base's parameters, then each module's parameters in registered order, deduplicating by `id` (first declaration wins). Then apply `parameterOverrides` — the country-level headline calibration — which replaces the `defaultValue` of each matched parameter without changing its bounds, group, or description.
2. **Variable union.** Same deduplication strategy. The state/control partition is preserved verbatim; the solver downstream assumes the ordering convention encoded in each module's variable list.
3. **Shock union and observable union.** Identical pattern. Every shock declared by an active module becomes a row in the IRF panel; every observable becomes a fan-chart column. The dashboard, exports, and diagnostics iterate these unions, so adding a module is sufficient to populate them — no view code changes.
4. **Equation list concatenation.** Equations are display-only in v1 (used in the Model view and printed exports). Each module contributes its own equation strings, which the compiled template joins in base + module order.

The seam this design produces — *composition resolves before the solver sees anything* — means the solver itself never needs to know which modules are active. It receives a flat list of parameters, variables, shocks, and observables, and runs the same simulation regardless of how that list was assembled.

Validation: incompatibility and requirements

Before compilation, the pipeline runs `DsgeModelConfiguration.validationError()` (see `ModelComposition.swift:200–217`). Three checks:

1. **Base required modules.** Each base declares a list of modules it requires. The Small Open Economy base requires `new_keynesian_core` and `monetary_policy`. The Currency Union base requires `new_keynesian_core` and `exogenous_monetary_rate`. Configurations that omit a required module fail validation with a human-readable error.
2. **Base incompatible modules.** Each base declares modules incompatible with it. The Currency Union base lists `monetary_policy` as incompatible, because allowing both `monetary_policy` and `exogenous_monetary_rate` would supply two policy-rate equations.

3. **Module-level compatibility.** Each module declares which bases it is compatible with (empty array = all). The Tourism and Commodity Exporter modules restrict themselves to Small Open Economy. Module-level required modules cover cross-module dependencies (every module currently requires the NK core).

These constraints are enforced at run time, not compile time, because the asset bundle is editable: a custom country file could in principle request an invalid configuration. Validation failure prevents the configuration from being passed to the solver, producing a "Configuration invalid for country X" message instead.

Country-level headline overrides

The `Country.headlineCalibration` dictionary supplies 5–8 numbers that replace the archetype defaults. These are the parameters that vary materially within an archetype:

PARAMETER	WHAT IT CONTROLS	NORWAY EXAMPLE	MALDIVES EXAMPLE
`alpha`	Trade openness — imports / consumption	0.32	0.85
`pi_ss`	Steady-state inflation target	0.005 (2 %/yr)	0.005 (2 %/yr)
`by_ss`	Steady-state debt-to-GDP	0.45	1.15
`phi_comm`	Commodity share of exports (if active)	0.50	n/a
`phi_tour`	Tourism share of GDP (if active)	n/a	0.35

The overrides apply at the compilation step (`ModelComposition.swift:144–161`), so the parameter that reaches the solver carries the country-specific value while keeping the bounds, group, and description of the archetype- level declaration.

Canonical-form hash

Every solved run carries a SHA-256 hash that uniquely identifies the parameter calibration that produced it. The hash is the basis of the vintage provenance system — two runs with the same hash are guaranteed by construction to have used identical inputs.

The canonical form serialises:

- `templateID` — the compiled template identifier, including the archetype and country (e.g. `soe_oil_exporter.no`).
- `templateVersion` — bumps when the template structure changes (new module added, equation form modified).
- The full parameter map, with keys sorted lexicographically and values formatted using `%.17g` (lossless decimal representation of the IEEE 754 double).
- `periods` — the simulation horizon at the time of the run.
- The list of active module IDs, sorted lexicographically.

The serialised string is then SHA-256'd. The first 6 hex characters are surfaced in the UI as the "run id short", which researchers paste in Slack and email to verify reproducibility ("did you run hash `a3f7c1` too?").

The choice of `%.17g` matters: it preserves bit-exact equality across platforms. A run produced on Apple Silicon and a run produced on x86 of the same model with the same parameters will produce the same hash, even though the floating-point representations of the simulation paths may differ in the last bit. This is essential for cross-platform reproducibility once the Windows port lands.

Vintage provenance

A vintage is a snapshot of a solved run, persisted as a `Vintage` SwiftData row (see `DSGEBuilder/Models/Persistence.swift`). Each vintage carries:

- `calibrationHash` — the canonical-form hash described above.
- `templateID`, `templateVersion` — for forward-compatible reading by future versions of the platform that may have evolved the template schema.
- `solverVersion` — pinned to the solver release that produced the run, so changes to the simulator after the fact don't silently invalidate saved figures.
- `platform` — operating system + CPU architecture, recorded for cross-platform reproducibility audits.
- `runStateJSON` — the entire canonical run state (parameters, IRFs, fan bands, moments, diagnostics, country metadata) serialised as JSON. This is the source of truth for every export — HTML, PDF, Quarto, LaTeX, Excel, CSV all read from this single payload.

The vintage is also mirrored to disk as `<container>/vintages/<uuid>.json` so the file is independently readable without SwiftData. External tooling (another econometrician's R script, a peer reviewer's Python check) can load the JSON directly and rebuild the same figures.

What this enables

The combination of declarative composition + canonical hashing + immutable vintage records gives the platform three properties that the audience cares about:

1. **Bit-exact reproducibility within a platform.** Same hash → same numbers, full stop.
2. **Cross-platform reproducibility.** Same hash on macOS and Windows ports produces the same hash because the canonical-form serialiser is platform-independent (`%.17g` representation, sorted keys, byte-exact SHA-256).
3. **Audit-grade provenance.** Every figure in an exported notebook ties back to the vintage hash printed on its cover. A peer reviewer can demand the hash, re-import the JSON into the platform, and verify the figure was not retouched after the fact.

The cost of these properties is the discipline of canonical-form serialisation. The benefit is that the platform produces research output that a referee will not need to re-derive in Dynare to trust.

Limitations of v1 provenance

The hash currently identifies *the parameter calibration*, not *the data*. If a user uploads a CSV through the Data tab and the fan-chart history overlay changes, the hash does not change — the hash describes the model input, not the visualisation input.

This is by design in v1: the solver does not yet consume uploaded data (no Kalman filter, no Bayesian estimation, no data-conditioned forecast). Once those features ship, the canonical form will be extended to include a content-hash of the dataset rows, so a Bayesian-estimated run can be reproduced bit-exactly only when both the calibration and the data match.

A second limitation: the modular architecture allows custom modules added via the customer asset path (see the business-model documentation). When a vintage uses a custom module, the canonical form records the module ID but not its source code. Reproducing such a vintage requires the customer's module bundle. v2 will add a content-hash of the active module bundle to the provenance manifest to make this explicit.

Annex A — Bibliography

References cited across the Engine SOP, alphabetised by first author.

An, S. and F. Schorfheide (2007). "Bayesian Analysis of DSGE Models." *Econometric Reviews*, 26(2–4): 113–172.

Bank of England (1996). *Practical Issues Arising From the Introduction of the Inflation Target.* Bank of England.

Blanchard, O. J. and C. M. Kahn (1980). "The Solution of Linear Difference Models under Rational Expectations." *Econometrica*, 48(5): 1305–1311.

Britton, E., P. Fisher and J. Whitley (1998). "The Inflation Report Projections: Understanding the Fan Chart." *Bank of England Quarterly Bulletin*, February: 30–37.

Fernández-Villaverde, J., J. Rubio-Ramírez and F. Schorfheide (2016). "Solution and Estimation Methods for DSGE Models." *Handbook of Macroeconomics*, vol. 2, ch. 9.

Galí, J. (2008). *Monetary Policy, Inflation, and the Business Cycle: An Introduction to the New Keynesian Framework.* Princeton University Press.

Galí, J. and T. Monacelli (2005). "Monetary Policy and Exchange Rate Volatility in a Small Open Economy." *Review of Economic Studies*, 72(3): 707–734.

Hamilton, J. D. (1994). *Time Series Analysis.* Princeton University Press.

Klein, P. (2000). "Using the Generalized Schur Form to Solve a Multivariate Linear Rational Expectations Model." *Journal of Economic Dynamics and Control*, 24(10): 1405–1423.

Pfeifer, J. (2014). *An Introduction to Graphs in Dynare.* Working paper, University of Mannheim.

Pinheiro, M. (2018). "Variance Saturation in BoE-Style Fan Charts." Working paper.

Sims, C. A. (2002). "Solving Linear Rational Expectations Models." *Computational Economics*, 20(1–2): 1–20.

If you spot a citation in the chapters that isn't listed here, raise it as a defect. The bibliography is load-bearing for the document's credibility.

Annex B — Symbol Glossary

Symbols used across the Engine SOP, grouped by where they appear.

Time and indexing

SYMBOL	MEANING
t	Calendar time (quarter or month, per the configuration's frequency)
h	Horizon — period offset relative to forecast start
$\hat{x}_t$	Deviation of variable x from its steady-state value at time t
$\mathbb{E}_t$	Conditional expectation given information at time t
ε_t	Vector of structural shocks at time t

State-space form

SYMBOL	MEANING
x_t	Full state vector — predetermined + jump variables
A	State-transition matrix
B	Shock-loading matrix
C	Observable-mapping matrix
Q	Shock variance-covariance matrix (typically diagonal)
Σ	Unconditional variance of the state

NK core parameters

SYMBOL	MEANING
β	Discount factor
σ	Inverse elasticity of intertemporal substitution
κ	Slope of the Phillips curve
ρ_x	Persistence of the output gap process
ρ_a	Persistence of productivity
ρ_d	Persistence of demand shock
π_{ss}	Steady-state inflation target

SYMBOL	MEANING
r_{ss}	Steady-state real interest rate

Monetary policy

SYMBOL	MEANING
ρ_i	Interest-rate smoothing parameter
ϕ_π	Taylor-rule inflation response
ϕ_y	Taylor-rule output-gap response
ρ_{iext}	AR(1) persistence of the exogenous policy rate (currency-union members)

Open-economy block (SOE base)

SYMBOL	MEANING
α	Trade openness — import share of consumption
λ	Open-economy Phillips-curve passthrough
q_t	Real exchange rate (deviation form)
y_t^*	Foreign output (deviation)
π_t^*	Foreign inflation (deviation)
ρ_{y^*}	Persistence of foreign output
ρ_{π^*}	Persistence of foreign inflation

Fan chart formula

SYMBOL	MEANING
$\mathrm{sd}(h)$	Standard deviation of the forecast at horizon h
$\bar{\sigma}$	Calibrated terminal forecast standard deviation per variable
τ	Saturation timescale of the variance formula ($\tau = 8$ quarters)
z_q	Gaussian quantile multiplier for credible region q

Module-specific

SYMBOL	MEANING
ϕ_{comm}	Commodity share of exports
η_{comm}	Pass-through of commodity price shock to terms of trade
ρ_{comm}	Persistence of world commodity price
ϕ_{tour}	Tourism share of GDP
η_{tour}	Tourism demand elasticity to foreign output
ρ_{tour}	Persistence of tourism shock
ϕ_{peg}	Strength of the peg anchor
χ_{ff}	Sensitivity of external finance premium to net worth
ϕ_{ff}	Pass-through of external finance premium into the IS curve
b_t	Government debt, scaled by GDP
ϕ_b	Fiscal response to debt deviations

Solver internals

SYMBOL	MEANING
`ShockChannel`	The state variable an impulse loads onto
`forecastFan`	Dictionary mapping variable id → array of `PercentileBand`
`perShockIRFs`	Nested dictionary `[shockID][variableID] = [Double]`
`terminalSD`	Per-variable calibrated terminal forecast $\bar{\sigma}$
`calibrationHash`	SHA-256 hash of the canonical-form serialisation (Chapter 05)