The Macroeconomic Effect of AI: Sizing the Software Engineering Channel*

Alex Blumenfeld **Jonathon Hazell** **Chen Lian** **Andreas Schaab**

September 14, 2026

Abstract

We measure how artificial intelligence (AI) affects the economy through its impact on software engineering productivity. We use information from financial markets to develop a forward-looking measure that is available in real time. We estimate the sensitivity of each firm's stock return to an AI stock market index, and how this sensitivity depends on the share of firm payroll in software engineering. We use a model to map this cross-sectional relationship into software engineering productivity gains. From November 2022 to December 2025, AI increased the market's expected present value of software engineering productivity by the equivalent of a permanent 32.6% productivity increase. The corresponding effect on the level of GDP is 3.6% in the baseline and 6.5% when higher software engineering productivity also raises R&D productivity. By mid-2026, amid rapid progress in coding agents, the effect of AI on productivity and GDP had more than doubled relative to the end of 2025.

Keywords: Artificial intelligence, software engineering, productivity, asset prices, economic growth.

*Blumenfeld: UC Berkeley; alex.blumenfeld@berkeley.edu. Hazell: London School of Economics; j.hazell@lse.ac.uk. Lian: UC Berkeley and NBER; chen_lian@berkeley.edu. Schaab: UC Berkeley and NBER; schaab@berkeley.edu. We thank Saleem Bahaj, Maarten De Ridder, Basil Halperin, Xavier Jaravel, Pete Klenow, David Lagakos, Eben Lazarus, Huiyu Li, Yueran Ma, Benjamin Moll, Christina Patterson, Lukasz Rachel, Christopher Tonetti and seminar participants at the NBER Summer Institute (Economic Growth and Long-Run Macroeconomic Development) and SED for valuable comments and suggestions; Gregor Schubert for sharing data; and Sven van Holten Charria and Nicholas Tokay for outstanding research assistance.

1 Introduction

How will artificial intelligence (AI) affect the economy? Forecasts of the effects on GDP span an enormous range: from modest gains of roughly one percent over a decade (Acemoglu, 2025) to scenarios in which AI raises growth rates by an order of magnitude (Trammell and Korinek, 2023; Davidson et al., 2026). This wide range points to the difficulty of measuring the impact of AI, in part because many of the advances have yet to play out.

A key channel through which AI affects the economy—both recently and likely in the future—is software engineering. Since the release of ChatGPT in November 2022, software engineering has become more productive, being a leading use case of AI. The pace of improvement seems to be accelerating: in 2026, coding agents such as Claude Code and Codex have revolutionized software engineering. More advances are likely in the future.

This paper measures the effects of AI on software engineering productivity and, through this channel, on the macroeconomy. We use information from financial markets. The main idea is that if AI raises the productivity of software engineers, then firms that rely more heavily on them should benefit more. Therefore news about AI generates higher expected profits for these firms. These profits are capitalized into larger stock price responses. Combining these stock price responses with a model, we can pin down the increase in software engineering productivity expected by financial markets.

The view from financial markets is unlikely to be perfectly accurate. However, a market-based approach has several significant advantages. First, market prices are forward-looking, providing a signal about advances in AI that are yet to come. Second, market prices are available at high frequency and in real time, which is critical given the rapid development and adoption of AI. Third, by using firm-level information from stock prices, one can measure overall software engineering productivity improvements, rather than gains on isolated tasks within the firm.

In the empirical part of the paper, we estimate how the sensitivity of firm stock returns to news about AI depends on the firm's share of payroll in software engineering. We use a two-stage approach. In the first stage, we estimate each firm's stock return sensitivity to returns on an AI stock market index. In particular, we regress firm-level stock returns on the return of the ROBO Global Artificial Intelligence index (THNQ), controlling for standard Fama-French risk factors. In the second stage, we regress the firm-level sensitivity to the AI index on firms' software engineering payroll shares, controlling for industry fixed effects and firm size. The payroll share is from Revelio Labs' employee-level data. We discuss how the second-stage slope isolates changes in software engineering productivity, even if movements in the AI index reflect shocks other than software engineering productivity (e.g. overall productivity shocks, discount rate shocks, and demand shocks that shift expenditure toward software-intensive goods).

To cleanly measure how AI raises software engineering productivity, we base our productivity estimate on differential responses across a sample of firms that use software engineering as an input, rather than firms that produce or sell software (e.g. so-called "SaaS" firms). We exclude firms in software-producing industries or the semiconductor supply chain. Firms that use software engineering intensively in our sample include: eBay (online marketplaces), Sonos (electronics manufacturing), and Expedia or Airbnb (travel booking platforms).

Our second-stage regression yields an estimated slope of 1.24. That is, if two firms' payroll shares differ by one percentage point, a one-percentage-point movement in the AI index predicts a return differential of 1.24 basis points. The result is robust across a wide range of specifications that control for alternative or additional risk factors, or use other AI stock market indexes, in the first stage; and for firms' product-market exposure to generative AI, standard financial ratios, and fine industry fixed effects in the second stage. A formal coefficient-stability test following Oster (2019) suggests unobserved heterogeneity does not explain the cross-sectional relationship. Year-by-year regressions are stable in every year from 2022 through 2025.

We combine our empirical estimates with a model to measure the expected present value of increases in software engineering productivity due to AI. The second-stage regression coefficient reflects two components: the key object κ, which measures the market's expected present value of increases in software engineering productivity per unit of excess return on the AI index (residualized against risk factors), and how firms' stock return sensitivity to software engineering productivity gains depends on differences in their software engineering payroll shares. The model pins down the latter in terms of a few key parameters, including the elasticity of substitution across goods and the substitution elasticity between software engineering and other labor.[1] The model then allows us to translate the second-stage slope into κ. Multiplying κ by the cumulative residualized excess AI-index return in a given window yields software engineering productivity gains due to AI news over that window.

From the launch of ChatGPT in November 2022, through December 2025, AI news led to a present-value increase in software engineering productivity equivalent to a permanent increase of 32.6%. This estimate is comparable in magnitude to the 21–56% task-level speed-ups found in the experimental literature (Peng et al., 2023; Paradis et al., 2025). Two offsetting forces can explain the similarity. On one hand, our present-value estimate includes expected future gains in software engineering productivity, which should exceed the gains from today's tools alone. On the other hand, complementarities and bottlenecks between tasks imply that aggregate productivity gains

[1]The baseline model features firms that produce differentiated goods, substitute between software engineers and other labor, and differ in the intensity with which they use software engineers. As we discuss, this simple model is isomorphic to a richer task-based model in which firms buy software from a sector that combines AI and software engineers.

should be smaller than task-level gains (Jones and Tonetti, 2026; Demirer et al., 2026).

One major benefit of our market-based approach is that it provides a high-frequency, near-real-time measure. We extend our measure through the first half of 2026 using the latest data. Expected software engineering productivity gains driven by news in the first half of 2026 were larger than those driven by news over the previous three years. This acceleration coincides with the rapid development and widespread adoption of coding agents such as Codex and Claude Code. Markets appear to expect significant productivity gains from these tools. Our series will provide a timely indicator of further productivity gains as the frontier continues to advance.

We next translate the software engineering productivity gains due to AI into an impact on GDP. Our baseline calculation applies a modified version of Hulten's theorem in the spirit of Aghion and Bunel (2024) and Acemoglu (2025). Because we focus on the aggregate impact, we also include firms that produce and sell software in this calculation; software engineering productivity gains lower their marginal costs as well. News about AI from November 2022 to December 2025 corresponds to a present-value GDP increase equivalent to a permanent 3.61 percent level increase.

We then consider two extensions and show that the software engineering productivity gain and GDP impact remain similar. First, we introduce a task-based framework with an explicit AI sector. Software suppliers bundle digital tasks performed by either software engineering labor or AI according to comparative advantage. Advances in AI replace software engineering labor and raise the productivity of software suppliers. Software suppliers then provide software inputs to differentiated-good producers. We derive an isomorphism between this model and our baseline framework. The mapping from the empirical estimates to the productivity gains and GDP impact is largely unchanged, though we require additional data on software spending. Second, we embed the framework in the full U.S. input-output network, explicitly capturing features such as upstream production and the economy-wide use of software. The network extension changes the inferred productivity gain and GDP impact only moderately.

We finally evaluate how the GDP impact changes when software engineering affects innovation. We embed our framework in a semi-endogenous growth model with an R&D sector, to which software engineering is an input. This extension leaves the mapping from the empirical estimates to software engineering productivity gains unchanged but significantly amplifies the mapping from these productivity gains to GDP. The reason is that higher software engineering productivity raises effective R&D input. Under our calibration, the implied impact on GDP nearly doubles due to software's effect on R&D. These results are large relative to the other model extensions we have considered.

Literature review. A growing literature quantifies the macroeconomic impact of AI. The starting point is often occupation-level AI exposure measures that classify tasks by their susceptibility

to automation (Eloundou et al., 2024; Acemoglu et al., 2022; Hampole et al., 2025). Papers then use these measures to quantify the macroeconomic effects, aggregating across various channels including the effect of AI on software engineering productivity. The estimates span a wide range (see also Karger et al., 2026 for expert disagreement about AI's macroeconomic effects). For example, Acemoglu (2025) estimates cumulative TFP level gains of 0.53–0.66% over a decade, while Aghion and Bunel (2024) estimate cumulative TFP level gains of 6.8–13% over a decade. We find markets expected TFP level gains, through the software engineering channel alone, of 3.61% at the end of 2025 and more than twice that amount by mid-2026.

Recent studies find large task-level speed-ups from AI coding assistants (e.g. Peng et al., 2023; Paradis et al., 2025; Cui et al., 2026; Gambacorta et al., 2024; Murphy-Hill et al., 2026). However, translating task-level gains into effects on firm-level or aggregate-level productivity is difficult (e.g. Brynjolfsson et al., 2021). Jones and Tonetti (2026) explain why: if production involves complementary tasks, then "weak links" attenuate the translation of task-level improvements from AI into firm-level or aggregate productivity gains. Demirer et al. (2026) provide direct evidence of this attenuation in software development. Large gains in coding activity translate into smaller increases in projects and releases and no detectable increase in consumer usage of new apps. Our paper takes a different route. Because asset prices already embed market beliefs about task-to-job aggregation, we sidestep these aggregation difficulties.

A growing literature uses asset prices to study how markets value the effects of AI on firms and the broader economy. In a pioneering paper, Eisfeldt et al. (2026) construct a novel measure of firms' exposure to generative AI, and show that firms with high exposure gained roughly 5% in market value relative to low-exposure firms after ChatGPT's release. The results are consistent with generative AI substituting for workers.[2] Andrews and Farboodi (2025) study the behavior of US bond yields around major AI release dates, and find clear evidence that investors price the possibility of transformative AI. Chow et al. (2026) argue more generally that long-term real interest rates provide an excellent way to assess whether markets price in transformative growth. Borri et al. (2026) construct a novel, high-frequency AI factor from OpenRouter token consumption and find that firms with greater exposure to the factor earn higher subsequent returns. We differ from these papers in focusing on the software engineering channel of AI, and in providing a framework to size the productivity gains using asset prices.

A related literature studies how AI affects firm-level outcomes. For instance, Babina et al. (2026) show that AI investments have led to productivity gains, driven by the accumulation of organization capital built by AI-skilled workers.[3] Hampole et al. (2025) construct novel measures of AI exposure,

[2] Bertomeu et al. (2025) use Italy's temporary ChatGPT ban and find that high-exposure firms lost roughly 6% in market value relative to low-exposure firms during the ban.

[3] Babina et al. (2024) show that firms investing in earlier waves of AI experience higher growth in sales, employment,

and find that AI substitutes for labor at the task level, but also generates productivity gains and task reallocation that dampen this substitution effect. Brynjolfsson et al. (2026) and Hosseini Maasoum and Lichtinger (2026) find early evidence consistent with AI leading to large declines in hiring for young and junior workers in exposed occupations. We complement these papers by providing not only evidence on the effects of AI to date but also a forward-looking assessment via financial markets.

Finally, our work connects to the theoretical literature on AI, automation, and growth. In particular, it builds on the task-based framework for analyzing how automation displaces labor (Acemoglu and Restrepo, 2018, 2019, 2022). Recent work emphasizes that AI-driven changes in task composition interact with workers' comparative advantage and occupational mobility to produce heterogeneous labor market effects (Freund and Mann, 2026; Althoff and Reichardt, 2026; Fan, 2025). Growth-theoretic models show that AI's long-run impact depends critically on whether it can automate the research process itself (Aghion et al., 2019; Korinek and Suh, 2024; Bontadini et al., 2026; Davidson et al., 2026), consistent with our findings on the importance of accounting for the innovation channel when sizing the GDP impact of AI.

Roadmap. Section 2 develops our methods. Section 3 presents the empirical implementation and estimates of software engineering productivity gains. Section 4 quantifies the GDP impact and considers extensions. Section 5 incorporates innovation, and Section 6 concludes. Main proofs appear at the end of the main text. The Supplemental Appendix contains additional tables, figures, data details, and proofs. The additional materials contain further empirical results and extensions.

2 Sizing the Software Engineering Productivity Gain from AI

We develop a framework that combines cross-sectional estimates with a general equilibrium model to infer software engineering (SWE) productivity changes due to AI. To simplify the exposition and derive closed-form expressions for the objects of interest, we use a stripped-down environment. Sections 4 and 5 later show that our procedure is robust to multiple extensions and explain how to translate the implied productivity change into a GDP impact.

2.1 Environment

Time is discrete, indexed by $t \in \{0, 1, \ldots\}$. The economy comprises a unit continuum of firms, indexed by $j \in [0, 1]$, each producing a distinct good. Each firm is small and takes aggregates as given. A representative household owns all firms, and financial markets are complete. There

and market value, in large part because AI helps them innovate their products.

are three aggregate shocks: S_t, A_t, and β_t. S_t denotes the productivity of software engineers. A_t denotes Hicks-neutral aggregate productivity. β_t is a time-varying one-period discount factor. The aggregate shocks may be arbitrarily correlated.

Households. The representative household is endowed with s units of software engineering labor and L units of non-SWE labor, both supplied inelastically. Their respective wages at date t are w_t and W_t. Household consumption preferences are

$$\mathbb{E}_0\Big[\sum_{t=0}^{\infty}\Big(\prod_{\ell=0}^{t-1}\beta_\ell\Big)u(C_t)\Big],$$

with the convention $\prod_{\ell=0}^{-1}=1$, where $\beta_t\in(0,1)$ and has steady-state value β. The consumption bundle is a standard CES aggregate with fixed demand weights ω_j integrating to one,

$$C_t=\Big[\int_0^1\omega_j^{1/\varepsilon}c_{jt}^{(\varepsilon-1)/\varepsilon}dj\Big]^{\varepsilon/(\varepsilon-1)},\qquad\int_0^1\omega_j dj=1.$$

Cost minimization implies

$$c_{jt}=\omega_j\Big(\frac{p_{jt}}{P_t}\Big)^{-\varepsilon}C_t\qquad\text{and}\qquad P_t=\Big[\int_0^1\omega_j p_{jt}^{1-\varepsilon}dj\Big]^{1/(1-\varepsilon)},$$

where P_t is the consumer price index and p_{jt} denotes the price of good j. The aggregate price index serves as numeraire, $P_t=1$ for all t. The stochastic discount factor (SDF) is $\Xi_{t,t+1}=\beta_t u_c(C_{t+1})/u_c(C_t)$.

Production. Each firm j produces output using a technology with constant returns to scale. A CES labor composite combines SWE and non-SWE labor,

$$H_{jt}=\Big[\phi_j^{1/\sigma}(S_t s_{jt})^{(\sigma-1)/\sigma}+(1-\phi_j)^{1/\sigma}L_{jt}^{(\sigma-1)/\sigma}\Big]^{\sigma/(\sigma-1)},$$

where s_{jt} and L_{jt} are firm j's employment of SWE and non-SWE labor, and $\sigma>0$ is the elasticity of substitution across the two types of labor. The parameter $\phi_j\in(0,1)$ governs software engineering intensity within the labor composite and varies across firms. S_t is software engineering productivity, which augments SWE labor. The firm then combines the labor composite with materials (in units of final goods) through a Cobb-Douglas function:

$$y_{jt}=A_t\Big(\frac{H_{jt}}{\alpha}\Big)^{\alpha}\Big(\frac{m_{jt}}{1-\alpha}\Big)^{1-\alpha},\tag{1}$$

where m_{jt} is firm j's use of final-good materials and α is the labor share of total costs. A_t is a Hicks-neutral aggregate productivity shock. The total cost of firm j is $\mathscr{C}_{jt}=w_t s_{jt}+W_t L_{jt}+m_{jt}$. Let q_{jt} denote the unit cost of the labor composite,

$$q_{jt}=\Big[\phi_j(w_t/S_t)^{1-\sigma}+(1-\phi_j)W_t^{1-\sigma}\Big]^{1/(1-\sigma)}.$$

With final-good materials priced at the numeraire, firm j's unit cost is $mc_{jt} = \frac{1}{A_t} q_{jt}^{\alpha}$, where w_t/S_t is the effective software engineering wage. Firms face CES demand and charge a constant markup $p_{jt} = \frac{\varepsilon}{\varepsilon-1} mc_{jt}$. Per-period profits are a constant fraction of nominal revenue, $\pi_{jt} = \frac{1}{\varepsilon} p_{jt} y_{jt}$.

We define the *software engineering channel* as the effects of AI that operate through a reduction in the effective cost of firms' software input. In the benchmark model, this reduction is driven by higher SWE productivity S_t, which lowers the effective SWE wage w_t/S_t.

Section 4.2 shows that the software engineering channel operates similarly in a richer setting, in which an upstream sector sells software services to final-goods producers. This software sector combines software engineers and AI input, supplied by an explicit AI sector, using a task-based production technology. The software engineering channel in this richer environment is isomorphic to its baseline counterpart: AI advances automate software engineering tasks, replace software engineers, and lower the price of software services for firms.

We define firm j's software engineering payroll share as $\gamma_{jt} = w_t s_{jt}/(w_t s_{jt} + W_t L_{jt})$. Since α is the labor share of total cost, the corresponding software engineering cost share is $\alpha\gamma_{jt} = (w_t s_{jt})/\mathscr{C}_{jt}$. The CES labor composite implies $\gamma_{jt} = \phi_j (w_t/(S_t q_{jt}))^{1-\sigma}$, which is strictly increasing in ϕ_j. The value of firm j is the present discounted value of its profit stream,

$$V_{jt} = \mathbb{E}_t\Big[\sum_{\tau=0}^{\infty} \Xi_{t,t+\tau} \pi_{j,t+\tau}\Big]. \tag{2}$$

Here $\Xi_{t,t} = 1$ and $\Xi_{t,t+\tau} = \prod_{\ell=0}^{\tau-1} \Xi_{t+\ell,t+\ell+1}$ for $\tau \geq 1$.

Market clearing. Goods market clearing requires the production of each differentiated good to equal household consumption demand plus firms' final-good materials demand for that good. Labor market clearing requires $\int_0^1 s_{jt} dj = s$ and $\int_0^1 L_{jt} dj = L$. We define firm j's final-expenditure share as $f_{jt} = p_{jt} c_{jt}/C_t = \omega_j (p_{jt}/P_t)^{1-\varepsilon}$, with $\int_0^1 f_{jt} dj = 1$. These are the weights the price index and demand system operate on. In this model they are also sales shares, $f_{jt} = p_{jt} y_{jt}/\int_0^1 p_{kt} y_{kt} dk$, and value-added shares, $f_{jt} = (p_{jt} y_{jt} - m_{jt})/\int_0^1 (p_{kt} y_{kt} - m_{kt}) dk$. The aggregate (final-expenditure-weighted) software engineering payroll share is $\gamma_t = \int_0^1 f_{jt} \gamma_{jt} dj$.

2.2 Response of Firm-Level Stock Returns to Aggregate Shocks

This subsection characterizes the response of firm-level stock returns to changes in software engineering productivity, as well as other aggregate shocks. These results pave the way for our main result, which uses firm-level stock returns to back out changes in software engineering productivity.

We characterize the first-order response of the economy to a perturbation $(d\log S_t, d\log A_t, d\log \beta_t)$. We start by characterizing the first-order response of firm profits. We drop t subscripts when referring to steady-state values. Given CES demand with constant markups, and since the

numeraire satisfies $P_t = 1$, per-period profits satisfy:

$$d\log\pi_{jt} = (1-\varepsilon)d\log mc_{jt} + d\log C_t.$$

By Shephard's lemma, the unit cost moves as a cost-share-weighted average of effective input prices, so that

$$d\log mc_{jt} = -d\log A_t + \alpha\gamma_j(d\log w_t - d\log S_t) + \alpha(1-\gamma_j)d\log W_t,$$

where materials are priced in units of the final good, whose price is the numeraire, and therefore do not contribute a separate price term. Software engineering productivity S_t enters by reducing the effective SWE wage $d\log w_t - d\log S_t$, with an impact proportional to the firm's SWE payroll share γ_j scaled by the labor cost share α. The constant price index ($d\log P_t = 0$) implies the final-expenditure-weighted average of unit cost changes satisfies $0 = -d\log A_t + \alpha\big[\gamma(d\log w_t - d\log S_t) + (1-\gamma)d\log W_t\big]$, where $\gamma = \int_0^1 f_j\gamma_j dj$ is the final-expenditure-weighted aggregate software engineering payroll share. Subtracting this zero from $d\log mc_{jt}$ isolates the relative cost change, $d\log mc_{jt} = -\alpha(\gamma_j - \gamma)[d\log S_t - (d\log w_t - d\log W_t)]$. Substituting the relative cost change into the profit equation yields:

$$d\log\pi_{jt} = -(\varepsilon-1)\alpha(\gamma_j-\gamma)\big(d\log w_t - d\log W_t - d\log S_t\big) + d\log C_t. \tag{3}$$

Equation (3) suggests a promising link between software engineering productivity and firm-level profits, which will form the core of our measurement strategy. The equation characterizes the response of profits in terms of a firm-specific component that depends on software engineering productivity, and an aggregate component. The first, firm-specific component contains the effective wage of software engineers $\big(d\log w_t - d\log W_t - d\log S_t\big)$, relative to other labor and accounting for software engineering productivity. The relative effective wage falls either because software engineering productivity $d\log S_t$ rises, or because relative wages $d\log w_t - d\log W_t$ fall. When the relative effective wage falls, firms with a higher-than-average reliance on software engineers ($\gamma_j > \gamma$) experience a relative marginal cost decline. Since demand is elastic ($\varepsilon > 1$), consumers substitute toward these newly cheaper goods, translating the cost advantage into profit gains. The second, aggregate component captures changes in aggregate demand $d\log C_t$ that do not generate cross-sectional variation. Nor do other aggregate shocks generate cross-sectional variation in profits: the aggregate productivity shock A_t drops out of relative costs because it scales all production frontiers uniformly; the time-varying discount factor β_t shifts the stochastic discount factor uniformly across firms.

We can further characterize how firm-level profits respond to software engineering productivity shocks by solving for the general equilibrium response of the relative effective wage. When S_t increases, software engineering labor becomes more abundant in efficiency units, so the relative

effective software engineering wage falls to clear the labor market. The size of this adjustment is

$$d\log w_t - d\log W_t - d\log S_t = -\frac{1}{\Sigma} d\log S_t, \qquad \text{with} \quad \Sigma = \sigma + \big[\alpha(\varepsilon-1) - (\sigma-1)\big]\frac{\sigma_\gamma^2}{\gamma(1-\gamma)},$$

where $\sigma_\gamma^2 = \int_0^1 f_j(\gamma_j - \gamma)^2 dj$ is the cross-sectional variance of software engineering payroll shares. Σ is the aggregate elasticity of substitution between software engineers and other labor, accounting for both within-firm substitution and reallocation between firms that use software engineers more or less intensively. A higher aggregate elasticity of substitution between software engineers and other labor Σ dampens the equilibrium wage response: when the economy can substitute more easily between the two types of labor, demand for software engineering labor is more elastic. Therefore, the relative effective wage of software engineering labor falls less in response to the increase in effective labor supply induced by higher software engineering productivity.

The aggregate elasticity of substitution Σ combines two channels through which the economy absorbs a software engineering productivity shock. The first is within-firm substitution, governed by σ: as σ rises, firms substitute more easily toward the now-cheaper software engineering labor, so aggregate substitutability Σ rises as well. The second is between-firm reallocation: consumers shift spending toward software-intensive firms whose prices fall by more. Holding payroll shares and σ fixed, a higher demand elasticity ε or a higher labor share α raises Σ by strengthening the reallocation channel. The cross-sectional variance σ_γ^2 enters because this reallocation channel operates only when firms differ in SWE intensity; if all firms had the same payroll share, Σ would equal σ. These terms and channels have already been emphasized in the context of capital-labor substitution (Oberfield and Raval, 2021).

Substituting this general equilibrium wage adjustment back into the expression for the response of profits, equation (3), yields an expression for the response of firm-level profits that depends only on the firm-level response to software engineering productivity—which varies in the cross-section with software engineering payroll share—and the firm-invariant response to aggregate demand. This expression is

$$d\log \pi_{jt} = \delta^S(\gamma_j - \gamma) d\log S_t + d\log C_t, \qquad \delta^S \equiv \alpha\frac{\varepsilon-1}{\Sigma}. \tag{4}$$

The structural gradient δ^S is the profit elasticity with respect to S_t per unit of the software engineering payroll share.

We next begin mapping the profit response into firm returns by aggregating (4) across future horizons. Log-linearizing firm value (2) and substituting in the response of profits implies:

$$d\log V_{jt} = \delta^S(\gamma_j - \gamma)(1-\beta)\mathbb{E}_t\Big[\sum_{k=0}^{\infty}\beta^k d\log S_{t+k}\Big] + (1-\beta)\mathbb{E}_t\Big[\sum_{k=0}^{\infty}\beta^k\big(d\log C_{t+k} + d\log \Xi_{t,t+k}\big)\Big], \tag{5}$$

where the second term is firm-invariant. To connect with the empirical strategy based on stock

returns, let $R_{jt} \equiv V_{jt}/V_{j,t-1} - 1 \approx d\log V_{jt} - d\log V_{j,t-1}$ denote firm j's realized return.[4]

We wish to link returns to news about the present value of software engineering productivity. We define this news as

$$\mathscr{S}_t \equiv (1-\beta)\big(\mathbb{E}_t - \mathbb{E}_{t-1}\big)\Big[\sum_{k=0}^{\infty} \beta^k d\log S_{t+k}\Big]. \tag{6}$$

The $1-\beta$ term is an annuitization factor which normalizes news about the present value of software engineering productivity. Under this normalization and under a random-walk process for $\log S_t$, $\mathscr{S}_t$ equals the period-t innovation in log software engineering productivity, $d\log S_t - \mathbb{E}_{t-1}[d\log S_t]$.

The following lemma links firm-level returns to software engineering productivity news, and is the key result of this subsection.

Lemma 1 (Firm Return Response)**.** *To first order, each firm j's unexpected period-t return is linked to the period-t news in normalized present-value software engineering productivity by:*

$$R_{jt} - \mathbb{E}_{t-1}[R_{jt}] = \delta^S \mathscr{S}_t \gamma_j + \mu_t, \tag{7}$$

where μ_t is firm-invariant, collecting unexpected responses common to all firms.

The key feature of (7) is that news about software engineering productivity affects firm returns differentially according to firms' software engineering payroll shares. The structural gradient δ^S governs how strongly this news translates into cross-sectional return differences per unit of the payroll share. In contrast, Hicks-neutral productivity and discount-rate shocks move all firm values proportionally and are therefore absorbed into μ_t. The next subsection shows how to combine a regression based on (7) with δ^S to recover AI-driven software engineering productivity news.

2.3 Backing Out Software Engineering Productivity Gains due to AI

We now come to the main use of the framework, which is to use the cross-sectional return differences to back out software engineering productivity gains due to AI. For this purpose, we use the excess return of the AI index as an informative signal about AI-related news and project software engineering productivity news onto this return. This projection allows us to infer software engineering productivity news conditional on the excess AI-index return:

$$\mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}_t^{\mathrm{AI}}] = \kappa \cdot \tilde{R}_t^{\mathrm{AI}}, \qquad \text{where} \qquad \kappa \equiv \frac{\mathrm{Cov}_t(\mathscr{S}_t, \tilde{R}_t^{\mathrm{AI}})}{\mathrm{Var}_t(\tilde{R}_t^{\mathrm{AI}})}, \tag{8}$$

where $\mathbb{E}^*[\cdot \mid \cdot]$ denotes the best linear predictor. $\tilde{R}_t^{\mathrm{AI}}$ is the residualized excess AI-index return, obtained by subtracting the risk-free rate from the return on the AI index, $R_t^{\mathrm{AI}} - R_t^f$, and then

[4]The empirical returns are standard CRSP holding-period total returns. The return defined here is instead constructed from cum-dividend value. The two differ by a dividend adjustment known at $t-1$. This adjustment drops out of the unexpected return in Lemma 1. Together with condition (ii) imposed below, this leaves the projection in (9) unchanged.

removing movements explained by a vector of risk factors F_t (e.g., the Fama-French market, SMB, and HML factors).[5] We assume that (i) $\mathbb{E}[\tilde{R}_t^{\text{AI}}] = 0$, so that the AI index earns no unconditional abnormal return relative to the risk factors, and (ii) $\text{Cov}_t(\mathbb{E}_{t-1}[R_{jt}] - R_t^f, \tilde{R}_t^{\text{AI}}) = 0$ for every firm j, so that the residualized AI-index return is uncorrelated with conditional risk premia. Together, these conditions capture the idea that the risk factors absorb the priced discount-rate variation in the AI-index return. Both conditions hold if $\tilde{R}_t^{\text{AI}}$ is unforecastable at $t-1$, i.e., $\mathbb{E}_{t-1}[\tilde{R}_t^{\text{AI}}] = 0$.

The key object of interest is κ, which measures news about software engineering productivity associated with a unit change in the residualized excess AI-index return $\tilde{R}_t^{\text{AI}}$. Multiplying κ by $\tilde{R}_t^{\text{AI}}$ therefore yields period-t news about software engineering productivity attributable to AI.

We infer κ by combining the model with a regression. Under the two assumptions above, projecting the firm-level return differences (7) in Lemma 1 onto $\tilde{R}_t^{\text{AI}}$ yields:

$$\mathbb{E}^*[R_{jt} \mid \tilde{R}_t^{\text{AI}}] = \delta^S \kappa \tilde{R}_t^{\text{AI}} \gamma_j + \tilde{\mu}_j + \tilde{\mu}_t, \tag{9}$$

where $\tilde{\mu}_j$ and $\tilde{\mu}_t$ are firm and time fixed effects, respectively. Equation (9) takes the form of a regression. Regressing the firm-level return R_{jt} on the interaction between the AI excess return $\tilde{R}_t^{\text{AI}}$ and the firm's software engineering payroll share γ_j, controlling for firm and time fixed effects, yields a coefficient of $\delta^S \kappa$. We can then combine the regression coefficient with the calibrated model-based structural gradient δ^S in (4) to recover the object of interest κ.

Empirically, to implement (9) and ensure that our procedure for recovering κ is robust to the extensions of the simple model (e.g., other drivers of movements in the AI index, as discussed in the next subsection), we use a two-step approach that allows for additional firm-level controls. In the first stage, we regress firm j's excess return $R_{jt} - R_t^f$ on the excess AI-index return, $R_t^{\text{AI}} - R_t^f$, and the control factors F_t to obtain its first-stage AI-index beta. By the Frisch–Waugh–Lovell theorem, firm j's first-stage AI-index beta equals $\beta_j^{\text{AI}} = \text{Cov}_t(R_{jt} - R_t^f, \tilde{R}_t^{\text{AI}})/\text{Var}_t(\tilde{R}_t^{\text{AI}})$. In the second stage, we regress each firm j's AI-index beta β_j^{AI} on its payroll share γ_j and a vector of firm-level controls X_j to obtain the second-stage coefficient $\delta = \text{Cov}_j(\beta_j^{\text{AI}}, \gamma_j \mid X_j)/\text{Var}_j(\gamma_j \mid X_j)$.

The proposition below, the main result of this subsection, shows that the second-stage coefficient δ in our two-stage procedure equals the regression coefficient $\delta^S \kappa$ in (9), so the procedure indeed implements (9). Combining the second-stage coefficient δ with the structural gradient δ^S then recovers the object of interest κ.

Proposition 1 (Financial-Market-Implied Software Engineering Productivity Change)**.** *The best linear predictor of the period-t news in normalized present-value software engineering productivity*

[5] Formally, let $R_t^{\text{AI}} - R_t^f = a^{\text{AI}} + \sum_i \psi_i^{\text{AI}} F_{it} + \epsilon_t^{\text{AI}}$ denote the projection of the excess AI-index return on risk factors, and define $\tilde{R}_t^{\text{AI}} = R_t^{\text{AI}} - R_t^f - \sum_i \psi_i^{\text{AI}} F_{it}$.

given the contemporaneous movement in the residualized AI index is

$$\mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}_t^{AI}] = \kappa \cdot \tilde{R}_t^{AI}, \qquad \textit{where} \quad \kappa = \frac{\delta}{\delta^S} = \frac{\Sigma}{\alpha(\varepsilon-1)} \cdot \delta. \tag{10}$$

This proposition summarizes how to construct a series measuring news about software engineering productivity due to AI. Using firm-level stock prices gives this series three advantages. First, the series is forward looking: it captures the present value of productivity gains, reflecting not only the effects of AI to date but also the effects that the market expects in the future. Second, the measure can be constructed at high frequency and in real time because asset-price data are readily available. Third, by using firm-level information, our measure of software engineering productivity sidesteps the difficulty of aggregating how AI affects the productivity of tasks within the firm, which is complicated by the presence of "bottleneck tasks" whose productivity is difficult for AI to improve (Jones and Tonetti, 2026).

To measure cumulative news about software engineering productivity due to AI over a given period, we cumulate our series. For instance, the cumulative AI news from the launch of ChatGPT in November 2022 to the end of 2025 is

$$\sum_{t=\text{'22 Nov}}^{\text{end '25}} \mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}_t^{\text{AI}}] = \kappa \cdot \sum_{t=\text{'22 Nov}}^{\text{end '25}} \tilde{R}_t^{\text{AI}}.$$

2.4 Identifying Software Engineering Channel

In our simple environment, shocks to software engineering productivity are the only aggregate shocks that move AI index returns and affect firms differently. We now depart from the baseline environment by introducing two additional drivers of AI index returns that affect firms differentially, and we discuss how our procedure nevertheless identifies the software engineering channel and recovers software engineering productivity gains due to AI. First, we introduce an aggregate shock that shifts expenditure toward software-intensive goods. Second, we introduce discount-rate news that affects firm values differently.

Demand composition. Let Z_t be a demand shifter. The consumption bundle becomes

$$C_t = \left[\int_0^1 \omega_j(Z_t)^{\frac{1}{\varepsilon}} c_{jt}^{\frac{\varepsilon-1}{\varepsilon}} \, dj \right]^{\frac{\varepsilon}{\varepsilon-1}},$$

where the CES demand weights $\omega_j(Z_t)$, normalized so that $\int_0^1 \omega_j(Z_t) dj = 1$, generalize the constant benchmark weights ω_j by letting the shifter Z_t reallocate expenditure across goods. Production is unchanged from equation (1). Now, the full set of aggregate shocks is (S_t, A_t, β_t, Z_t).

The demand shifter Z_t can contaminate the payroll-share slope δ through two channels. First, there is a direct demand channel. Suppose Z_t shifts expenditure weights $\omega_j(Z_t)$ in a way that, conditional on regression controls, correlates with the software engineering payroll share γ_j. Then

the profits of software-intensive firms go up even if their costs do not change, because of higher demand. Second, there is a factor-price channel: if Z_t shifts demand toward software-intensive goods in a final-expenditure-weighted sense, demand for software engineering labor rises, driving up the relative wage of software engineers and generating cost differentials proportional to the software engineering payroll share γ_j. Assumption 1 shuts down both channels. We will verify both conditions of this assumption empirically, for example, by using product-market exposure to generative AI as a proxy for the demand-weight elasticity ω'_j.

Assumption 1. *The demand-weight elasticities $\omega'_j = \partial \log \omega_j / \partial \log Z|_{ss}$ have zero covariance with software engineering payroll shares in two distinct senses,*[6]

$$Cov_j(\omega'_j, \gamma_j \mid X_j) = 0 \qquad \textit{and} \qquad Cov_{f,j}(\omega'_j, \gamma_j) = 0.$$

The first condition targets the direct demand channel: the demand shift may raise the profits of high-γ_j firms simply because consumers now want their products more. The condition is conditional on firm-level controls X_j, consistent with our empirical implementation. We condition on X_j because only demand variation that survives the controls can bias the regression. When X_j includes industry fixed effects, the regression already absorbs any demand reallocation that merely moves spending across industries, so the condition asks only that no reallocation survive within industries, conditional on any additional firm-level controls in X_j.

The second condition targets the factor-price channel. When the final-expenditure-weighted covariance is zero, demand reallocation does not raise aggregate demand for software engineering labor, so the relative wage of software engineers does not move and no cost differentials arise. These weights appear here, and not in the first condition, because the labor market clears in weighted terms: each firm's demand for software engineers scales with its sales (equivalently, its final-expenditure share). Therefore large firms contribute more to aggregate software engineer demand. The wage responds to the final-expenditure-weighted shift of demand toward software-intensive firms. Later on we will provide an empirical test of this assumption.

Discount-rate heterogeneity. In the benchmark, the time-varying discount factor β_t shifts all firm valuations proportionally and is absorbed into the common component μ_t. In practice, firms' values may respond heterogeneously to common discount-rate news. If that heterogeneity covaries with the payroll share γ_j conditional on firm-level controls, and the residualized AI index captures discount-rate news that is not spanned by the first-stage factors, the estimated slope δ could conflate the software engineering productivity channel with discount-rate-driven valuation effects.

[6] We define $\text{Cov}_{f,j}(\omega'_j, \gamma_j) \equiv \int_0^1 f_j(\omega'_j - \bar{\omega}'_f)(\gamma_j - \gamma)dj$, where $\bar{\omega}'_f = \int_0^1 f_j \omega'_j dj$ and $\gamma = \int_0^1 f_j \gamma_j dj$. Since the final-expenditure weights integrate to one, this equals $\int_0^1 f_j(\gamma_j - \gamma)\omega'_j dj$.

We extend the firm-return equation (7) to allow for heterogeneous exposure to discount-rate news:

$$R_{jt} - \mathbb{E}_{t-1}[R_{jt}] = \delta^S \gamma_j \mathscr{S}_t + \eta_j \mathscr{D}_t + \mu_t, \tag{11}$$

where $\mathscr{D}_t$ is the period-t discount-rate news in normalized present-value terms, as defined in equation (F.4). The equation says that the firm return responds to software engineering productivity news through $\delta^S \gamma_j$ (as before) and to common SDF news through the duration loading η_j. Appendix F microfounds η_j through firms' franchise duration. Under standard AR(1) discount-rate news, η_j is strictly increasing in duration, reaching the benchmark value $\eta_j = 1$ for a perpetual claim.

Recall that $\tilde{R}_t^{\text{AI}}$ is obtained by residualizing the excess return on the AI stock market index against risk factors. These factors control for the variation in $\mathscr{D}_t$ that they span. Therefore, only heterogeneous exposure to discount-rate news not absorbed by these factors can bias the second-stage regression coefficient δ. In particular, $\delta = \kappa \delta^S + \kappa^{\Xi} \delta^{\Xi}$, where $\delta^{\Xi} \equiv \text{Cov}_j(\eta_j, \gamma_j \mid X_j)/\text{Var}_j(\gamma_j \mid X_j)$ and $\kappa^{\Xi} \equiv \text{Cov}_t(\mathscr{D}_t, \tilde{R}_t^{\text{AI}})/\text{Var}_t(\tilde{R}_t^{\text{AI}})$. The slope δ^{Ξ} measures whether software engineering-intensive firms have systematically different equity duration, while κ^{Ξ} measures whether the residualized AI index still loads on common discount-rate news. Thus, the bias $\kappa^{\Xi}\delta^{\Xi}$ is nonzero only if equity duration varies systematically with software engineering payroll share and the relevant discount-rate news is not absorbed by the risk factors. Assumption 2 requires that at least one of these ingredients is absent.

Assumption 2. *At least one of the following conditions holds:*

$$Cov_t(\mathscr{D}_t, \tilde{R}_t^{AI}) = 0 \qquad or \qquad Cov_j(\eta_j, \gamma_j \mid X_j) = 0.$$

The first condition (factor spanning) requires that the first-stage factors absorb the priced discount-rate variation in the AI index. The second condition (cross-sectional orthogonality) requires that discount-rate loadings are conditionally uncorrelated with software engineering payroll shares.

The factor spanning condition is strengthened by including richer controls in the first-stage regression (13). The baseline Fama-French three factors already absorb the component of discount-rate news spanned by broad market, size, and value-factor returns. In Section 3.3, we show that the regression coefficient δ remains stable after adding the NASDAQ 100 return (QQQ), the Fama-French five factors, and momentum to the first stage, supporting the first condition.[7]

The cross-sectional orthogonality condition is strengthened by including proxies for equity duration in the second-stage controls. In Section 3.3, we show that the regression coefficient δ is

[7] Babina et al. (2025) show that firms' AI investment leads to heterogeneous exposure to the market factor. Because the market factor is always controlled for in our first-stage regression, the heterogeneous exposure they study does not bias our estimates of δ.

stable when the second stage controls for book-to-market, operating profitability, investment, and leverage, standard cross-sectional pricing controls that proxy for equity duration (e.g., Lettau and Wachter, 2007; Campbell et al., 2010), supporting the second condition.

Proposition 2 (Identification)**.** *Consider the demand-composition extension under Assumption 1 and the discount-rate-heterogeneity extension under Assumption 2. In either case,*

$$\kappa = \frac{\delta}{\delta^S} = \frac{\Sigma}{\alpha(\varepsilon - 1)} \cdot \delta \tag{12}$$

and Proposition 1 continues to hold.

The separate discussion of these two threats illustrates a broader advantage of our procedure. The residualized excess return on the AI index $\tilde{R}_t^{\text{AI}}$ need not be a pure measure of news about AI-driven software engineering productivity. To back out software engineering productivity, we require only that $\tilde{R}_t^{\text{AI}}$ be a noisy signal of the software engineering productivity news $\mathscr{S}_t$. In each extension, $\tilde{R}_t^{\text{AI}}$ may reflect the corresponding demand or discount-rate news, as discussed above, or other news unrelated to software engineering productivity (e.g. Hicks-neutral productivity shocks). Any such component will not bias our method if firms' heterogeneous exposures to these shocks are uncorrelated with their software engineering payroll shares, conditional on firm-level controls.

3 Estimates of the Software Engineering Channel

This section implements the strategy of Section 2 to back out software engineering productivity gains due to AI. Section 3.1 describes the data sources and the construction of the analysis sample. Section 3.2 describes the main specification. Section 3.3 presents the baseline estimates of the cross-sectional slope δ in our two-step procedure. Section 3.4 uses the results from Section 2 to translate this slope into the implied software engineering productivity gain. It also shows how our approach can be used to update the market view, in real time.

3.1 Data

Our core data sources are resume history data from Revelio Labs, financial statement variables from Compustat, historical data on the Fama-French factors from Kenneth French's website, stock and exchange-traded fund (ETF) price data from the Center for Research in Security Prices (CRSP), and returns on AI-related ETFs that serve as our aggregate AI index.

Revelio Labs aggregates publicly available professional profiles from platforms such as LinkedIn to construct a comprehensive, worker-level dataset. Each record includes a standardized occupation classification that can be linked to O*NET codes, which Revelio then aggregates into approxi-

mately ten broad role categories (e.g. "Software Engineer"). Each record also includes a measure of total compensation imputed by Revelio using features of the job. The dataset also provides sampling weights that adjust for the fact that workers in some occupations are more likely to have public professional profiles than others. We use these data to construct the key firm-level regressor in our analysis: the share of total payroll attributable to software engineers.[8]

Compustat provides standardized financial statement variables drawn from firms' annual and quarterly public filings with the SEC. We use the Compustat annual fundamentals file to obtain firm-level measures of total assets, net sales, operating profitability, investment, leverage, and book-to-market ratios. We also use the Compustat-CRSP link table to merge accounting data with stock return data at the firm level.

The Fama-French factor returns, available from Kenneth French's data library, provide daily and weekly time series of the standard risk factors used in asset pricing (Fama and French, 1993, 2015). In our baseline results, we use the three factors of Fama and French (1993): the excess market return (MKT), the size factor (SMB), and the value factor (HML). In robustness exercises, we also include the investment factor (CMA) and the profitability factor (RMW) from the five-factor model of Fama and French (2015). These factors are designed to capture exposure to broad market forces and to well-documented cross-sectional return premia.[9]

The Center for Research in Security Prices (CRSP) US Stock and US Index databases provide historical security-level data on prices, returns, shares outstanding, and market capitalization for all securities listed on major US exchanges. We use weekly holding-period returns from CRSP for the firm-level time-series regressions in our first stage. CRSP also provides returns on exchange-traded funds, which we use to obtain the return series for the AI-related ETFs that serve as our aggregate AI index.

In the main results, we use THNQ as our AI stock market index. THNQ tracks the ROBO Global Artificial Intelligence Index, which selects constituents based on their revenue exposure to artificial intelligence and classifies them into two broad categories: firms providing AI infrastructure (such as cloud computing) and firms developing or deploying AI applications (such as healthcare AI). The index uses a modified equal-weight methodology, in which holdings are assigned to scoring tiers based on the depth of their AI revenue exposure and then approximately equally weighted within each tier, giving higher aggregate weight to firms with more direct AI exposure. The index

[8]Appendix Figure B.1 compares the occupation distribution in Revelio with that in representative data from the Occupational Employment Statistics; even with Revelio's sampling weights, our main dataset over-represents the "Management" and "Computer and Mathematical" occupation groups. We therefore present robustness tests addressing representativeness.

[9]MKT is the return on the US equity market minus the risk-free rate. SMB is the difference in returns between portfolios of small-cap and large-cap stocks. HML is the difference in returns between high book-to-market (value) firms and low book-to-market (growth) firms. CMA is the difference in returns between firms with low and high asset growth rates. RMW is the difference in returns between highly profitable and less profitable firms.

typically contains around 50 to 60 holdings and is rebalanced quarterly. Table E.1 lists the twenty largest holdings and their weights in the index. In our robustness checks, we replace THNQ with the AIQ index, which uses market-capitalization weights and has broader coverage.[10]

Our main measure of the importance of software engineers at a firm is the share of total compensation in Revelio that goes to all workers in the "Software Engineer" category in 2021 (the year before our stock-price panel begins).[11] We compute the total number of workers in each occupation at a firm using Revelio's individual-level weights. Since compensation is imputed, we will also report results using raw employment shares of software engineers rather than payroll shares.

Sample selection. Table A.2 shows how we arrive at our main analysis sample. The time period is from the start of 2022 to the end of 2025. We start with the initial CRSP-Compustat merge (Row 1). In Row 2, we restrict to nonfinancial firms by dropping firms with SIC codes 6000-6999. In Row 3, we retain only firms that match to Revelio in 2021. In Row 4, we compute average assets and average sales from 2017 to 2021, ignoring missing years, then restrict to firms with strictly positive average assets and sales over this period. In Row 5, we restrict to a balanced panel of firms with stock price data for all weeks of 2022 through 2025, and we also require non-missing market capitalization data for the end of 2021. In Row 6, we drop all firms that are among the holdings of either THNQ or AIQ as of March 2026.

To cleanly measure how AI raises software engineering productivity by reducing the cost of software engineering, we base our productivity estimate on a sample of firms that use software engineering as an input, excluding those that produce or sell software or inputs into AI production. In Row 7, we drop from our baseline sample a list of firms classified by GPT 5.6 Sol as being involved in the development or production of semiconductors. These firms' stock returns will likely covary with returns on an AI ETF because they supply hardware required to train and to run AI models.[12]

In Row 8, we drop firms that produce or sell software: as in Acemoglu et al. (2022), we exclude firms in NAICS sectors 51 (Information) and 54 (Professional, Scientific, and Technical Services), as many firms in these sectors are producers of AI rather than users. Finally, in Row 9, we drop the bottom 5% of remaining firms ranked by total employment in Revelio: sample selection is a concern for these firms, because they may have few resumes posted online. At this point, we have

[10] AIQ tracks the Indxx Artificial Intelligence and Big Data Index, which selects firms that derive a meaningful share of their revenue from AI, big data, or related technology segments; and uses market-capitalization weights capped at 3% at each rebalance. Table E.2 lists the twenty largest holdings and their index weights.

[11] Occupations in Revelio's "Software Engineer" category are linked to 125 distinct O*NET occupation codes. Table A.1 lists the twenty largest by compensation share.

[12] Specifically, we ask GPT 5.6 Sol to identify firms that are part of the semiconductor supply chain, based on their NAICS codes and their business model descriptions from Compustat. The majority of the 64 firms in our final list are in NAICS 334413 (Semiconductor and Related Device Manufacturing). Table E.3 summarizes the full set of dropped semiconductor-related firms, and Appendix C.5 contains the full text of the prompt used for the classification.

the final dataset. The table also lists the percentiles of 2021 sales. Even as we make the sample restrictions, the distribution of firm sales remains reasonably similar.[13]

We present descriptive statistics about our main regressor, the firm-level payroll share of software engineers. Table E.4 shows the top 20 firms with the highest software engineering intensity according to our measure. There seem to be two business models in common among the firms with a high share of software engineers, namely high-tech manufacturing (e.g. Sonos, Arlo) and e-commerce platforms (e.g. Expedia).[14]

3.2 Main Specification

To implement the firm-level regression (9) from the previous section, we adopt a two-stage procedure. In the first stage, in the spirit of Fama and MacBeth (1973), we obtain a firm-level AI news sensitivity (or "beta") by estimating each firm's stock return sensitivity to an AI index while controlling for standard risk factors. In the second stage, we regress these estimated betas on firms' software engineer payroll shares, controlling for industry fixed effects and other firm-level controls, to estimate the cross-sectional slope δ.[15]

First stage: firm-level AI betas. We start by estimating each firm's beta with respect to the return of an AI-focused exchange-traded fund. For each firm j in the regression sample, which we described in Section 3.1, we run the following time-series regression using weekly data:

$$R_{jt} - R_t^f = a_j + \beta_j^{\text{AI}}\left(R_t^{\text{AI}} - R_t^f\right) + \sum_i \psi_{ji} F_{it} + \epsilon_{jt}, \tag{13}$$

where R_{jt} is firm j's stock return, R_t^{AI} is the return on the AI index, R_t^f is the risk-free rate, and F_{it} are risk factors. In the baseline specification, the controls are the Fama-French three factors (Fama and French, 1993). The coefficient β_j^{AI} measures firm j's AI-factor loading: a one percentage point increase in the AI index excess return is associated with a β_j^{AI} percentage point movement in the firm's excess return. The risk factors absorb each stock's exposure to broad market forces, such as the market risk premium, size, and value.

Second stage: explaining cross-sectional betas. The cross-sectional model relates estimated firm betas to software-engineering intensity:

$$\hat{\beta}_j^{\text{AI}} = \delta \cdot \gamma_j + \Gamma \cdot X_j + u_j, \tag{14}$$

[13] Figure B.5 shows the distribution of firms across NAICS2 sectors after basic Compustat cleaning and in our cleaned final sample; the distributions are similar in each. In addition, Table A.3 presents medians of several firm characteristics for the firms in our final sample with the highest and lowest software engineer payroll shares.

[14] Figure B.6 shows the full distribution of software engineer payroll shares; for reference, in our main sample, there are 185 firms with zero software engineers observed in our dataset in 2021.

[15] By the Frisch-Waugh-Lovell theorem, this two-stage procedure is equivalent to regression equation (9).

where $\hat{\beta}_j^{\text{AI}}$ is the first-stage estimate of firm j's AI beta, γ_j is the software engineering payroll share, and X_j denotes the full vector of second-stage controls. In the baseline specification, X_j includes firm size controls (indicators for quintiles of average 2017–2021 sales and assets) together with a full set of industry indicators (either NAICS2 or NAICS3). We trim observations above the 99th percentile of the software engineering payroll share in our main analysis sample, which is 31.6%. The coefficient δ measures the increase in the AI beta per unit increase in a firm's software engineering payroll share, conditional on this common control set X_j. In return units, comparing two firms whose payroll shares differ by one percentage point, a one percentage point movement in the AI index predicts a return differential of δ basis points. The baseline regression is unweighted, with heteroskedasticity-robust standard errors. As discussed, to cleanly estimate software engineering productivity gains due to AI, we focus on firms that use software as an input into production. The sample therefore excludes firms that produce or sell software and firms in the semiconductor supply chain. However, when we use the model in Section 4 to infer the macroeconomic impact of AI through the software engineering channel, we account for these firms as well.

3.3 Baseline Estimates

This subsection presents our main empirical results. Table 1 reports estimates of the second-stage regression coefficient δ from (14). In the first column, we present estimates of the regression of the AI index beta on the software engineering payroll share without controls, which yields a coefficient of 1.70. The estimates remain similar as we add size controls in column 2 (1.81) and 2-digit industry fixed effects in column 3 (1.91). Column 4, with size controls and 3-digit industry fixed effects, is our baseline specification and yields $\delta = 1.24$. The coefficient means that, comparing two firms that differ by 1 pp in software engineering payroll share, a 1 pp AI index increase predicts a return that is 1.24 basis points higher for the more SWE-intensive firm. The estimates are statistically significant and precisely estimated in all specifications. In the final column, we add a control for workforce exposure to generative AI, measured as the employment-weighted share of a firm's occupational tasks that can be performed or assisted by large language models (Eisfeldt et al., 2026). The estimate is essentially unchanged at 1.28. We visualize the result with a binned scatter plot. Specifically, Figure 1 plots 40 bins of the software engineering payroll share on the x-axis, against the firm's sensitivity to the AI index on the y-axis, after controlling for NAICS3 and size fixed effects as well as generative AI task exposure (i.e. the specification in column 5 of the table). There is a clear positive relationship between software engineering payroll share and sensitivity to the AI

Table 1: Regression of THNQ beta on software engineer payroll share

Dependent Variable:	Beta on THNQ excess return				
	(1)	(2)	(3)	(4)	(5)
Software engineer payroll share	1.70^{***}	1.81^{***}	1.91^{***}	1.24^{***}	1.28^{***}
	(0.21)	(0.20)	(0.21)	(0.26)	(0.33)
Quintile of average 2017-2021 assets		Yes	Yes	Yes	Yes
Quintile of average 2017-2021 sales		Yes	Yes	Yes	Yes
NAICS2			Yes		
NAICS3				Yes	Yes
Quintile of GenAI task exposure (Eisfeldt et al., 2026)					Yes
Observations	1,855	1,855	1,855	1,850	1,144

Notes: The dependent variable is the firm-level beta on THNQ excess returns, estimated in a first-stage weekly regression of firm excess returns on THNQ excess returns with FF3 controls over the 2022–2025 balanced panel. The key regressor is the software engineer payroll share, measured using Revelio and defined as the share of firm payroll attributable to software engineer occupations. Columns differ only in the controls and fixed effects indicated in the table; specifications with fixed effects drop singleton fixed-effect groups if they are present. All columns trim observations above the 99th percentile of the software engineer payroll share in the main analysis sample. Heteroskedasticity-robust standard errors are reported in parentheses. Significance levels are denoted by $^{***}p < 0.01$, $^{**}p < 0.05$, and $^{*}p < 0.10$.

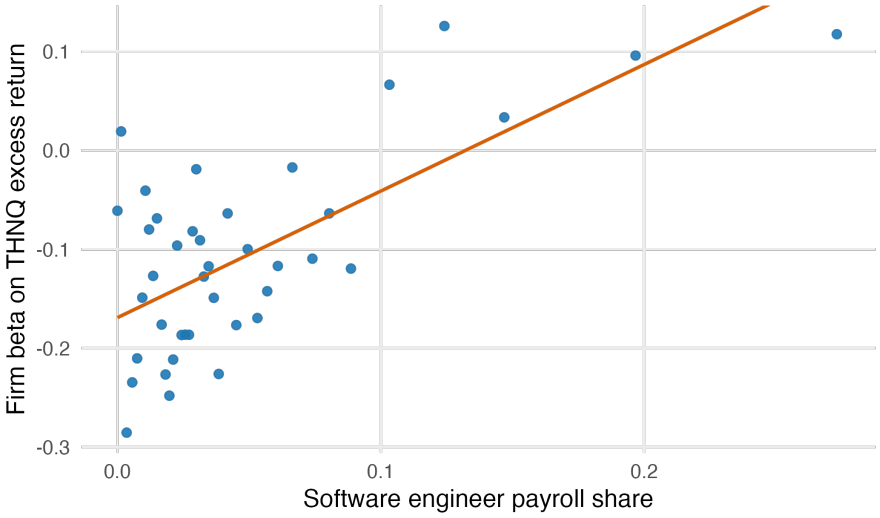
index. [16]

The estimated coefficients remain similar as we consider an extensive range of robustness exercises, as Table 2 documents. The first row is the baseline specification, from column 4 of Table 1. We start with robustness tests that speak directly to the two identification assumptions of Section 2.4, which require that other aggregate shocks do not generate cross-sectional variation in AI betas correlated with the software engineering payroll share.

The first threat is demand composition (Assumption 1). With industry fixed effects in the second stage, the direct demand channel (a violation of part 1 of Assumption 1) operates only through within-industry reallocation toward software-engineering-intensive firms. Rows 2–3 provide evidence against this channel by controlling for the most granular possible industry fixed effects, at the 4- and 6-digit levels. Row 4 further probes this channel by controlling for three measures of firms' exposure to generative AI through the product market, exactly as constructed in Eisfeldt et al. (2026): (1) an indicator for whether GPT classified a firm as a likely beneficiary of GenAI based on the business description section of its 10-K, (2) the number of AI-related keywords in its 10-K, and (3) the share of its workers with AI-related skills as defined in Babina et al. (2024). The coefficient remains stable, supporting part 1 of Assumption 1. The factor-price channel

[16] Figure B.7 shows the distribution of our first-stage betas, and Table E.5 shows summary statistics. This measure of firm-level loadings on our chosen AI index is centered near zero, and 18.2% of firms have AI betas that are significant at the 5% level, using Newey-West standard errors with four lags.

Figure 1: Software engineering payroll share vs. THNQ beta



Notes: The figure plots 40 equal-sized bins of the software engineering payroll share (x-axis) against the firm's estimated beta on THNQ excess returns (y-axis). We residualize against NAICS3 industry and firm size quintile fixed effects, as well as quintiles of the generative AI task exposure measure of Eisfeldt et al. (2026). The fitted line is the OLS slope from the corresponding column of Table 1.

(a violation of part 2 of Assumption 1) is probed separately by the final-expenditure-weighted covariance test discussed in Section 2.4. The test implemented in Appendix C.1 cannot reject the null that the final-expenditure-weighted covariance is zero, supporting part 2 of Assumption 1.

The second threat is discount-rate heterogeneity (Assumption 2), which biases δ only if the residualized AI index loads on discount-rate news not spanned by the first-stage factors and loadings covary with the SWE payroll share. Rows 5–6 study the spanning condition (part 1 of Assumption 2) by enriching the first-stage factor span with QQQ and additional Fama-French factors. Row 7 adds the momentum factor on top of the standard Fama-French three factors. Row 8 tests the cross-sectional orthogonality condition (part 2 of Assumption 2) by controlling for standard equity-duration proxies, namely book-to-market, operating profitability, investment, and leverage, in the second stage (e.g., Lettau and Wachter, 2007; Campbell et al., 2010). The stability of δ across these specifications supports both parts of Assumption 2.

Next, we add additional controls to the second-stage cross-sectional regression. Row 9 controls for baseline firm market capitalization. Row 10 controls for region, via the location of the firm's headquarters. Row 11 controls for the payroll shares of several other occupations, namely Engineer, Finance, Project and IT Specialist, and Technician. Row 12 controls for the firm's average compensation per worker.

Although, as discussed above, our methods allow the AI index to reflect shocks unrelated to AI, one concern is that such shocks may dominate its movements. As a prelude, Row 13 runs the first-stage regression (13) using daily rather than weekly data. Then, Row 14 assuages the concern by restricting the daily first-stage sample to the day of each major AI model release and the following day, when movements in the AI index are especially likely to reflect AI-related news. We use release dates for models from OpenAI, Anthropic, and Google between 2022 and 2025 compiled by Andrews and Farboodi (2025).

Finally, we add in assorted additional robustness tests. In row 15, we replace the standard Fama-French market factor in our first-stage regressions with an alternate version that excludes the "Magnificent 7" tech firms, to construct a market return less affected by AI-related news. Row 16 drops micro-cap firms, defined as firms with market capitalization below $1 billion at the end of 2021. Row 17 uses AIQ as the "AI index" as an alternative to our baseline choice of THNQ. Row 18 weights the regression by size (end of 2021 market capitalization). Since compensation in Revelio is imputed, row 19 instead uses raw employment shares of software engineers. Row 20 attempts to correct for selection in which types of workers are more likely to have online profiles that are represented in the Revelio data, by reweighting broad occupations to match the distribution of employment across occupations in representative survey data; we describe the full procedure in Appendix C.3. Row 21 assesses robustness to potential measurement error in the software engineer payroll share by instrumenting its 2021 value with its 2020 value. Finally, row 22 addresses potential concerns about serial correlation in the first-stage time series and sampling uncertainty in the cross section by reporting bootstrap standard errors.[17]

It is reassuring that the coefficients are stable as we consider various permutations. As Oster (2019) points out, stability of the coefficients as we add observed heterogeneity to the controls is informative. It suggests that unobserved heterogeneity is unlikely to bias the estimates. Building on this idea, we implement the formal test of Oster (2019) in Appendix C.2 and find that there is limited scope for omitted variable bias to explain our results.

Our baseline regression uses the software engineering payroll share as a continuous regressor. We now discretize firms with positive payroll shares into five quintiles. Table A.4 reports the results. As the table shows, relative to zero-share firms, AI sensitivity is higher in the top quintile and to some extent the fourth quintile. That is, the top tail of software engineering intensity drives the results. In Table A.5, we show this result is not driven by outliers, either at the top end of the software engineer payroll share distribution or at the bottom (as is apparent from Figure 1). We also study the stability of our estimates, by estimating δ separately for every year. Figure B.2 plots

[17] Calendar weeks are resampled using a moving-block bootstrap with block length 6 weeks, firms are resampled with replacement, and both stages are re-estimated in each iteration. The bootstrap standard error is larger than the baseline heteroskedasticity-robust standard error (0.47 vs. 0.26), but the coefficient remains statistically significant.

Table 2: Streamlined robustness checks relative to the NAICS3 FE anchor specification.

		Coefficient	Std. Error	Observations
1.	Baseline	1.24	0.26	1850
2.	NAICS4 FE	1.10	0.31	1768
3.	NAICS6 FE	0.75	0.40	1486
4.	Control for GenAI product exposure	1.41	0.38	837
5.	NASDAQ control in first stage	1.20	0.26	1850
6.	FF5 factors in first stage	1.04	0.28	1850
7.	Add momentum factor in first stage	1.12	0.26	1850
8.	Control for common financial ratios	1.45	0.27	1777
9.	Control for log 2021 market cap	1.38	0.26	1850
10.	HQ state FE	1.14	0.28	1702
11.	Control for adjacent occupation shares	1.42	0.30	1850
12.	Control for average compensation	1.30	0.26	1850
13.	Daily data in first stage	1.50	0.19	1850
14.	AI model release dates in first stage	1.71	0.55	1850
15.	Ex-"Mag 7" market factor in first stage	1.31	0.25	1850
16.	Drop micro-cap firms	1.67	0.27	1054
17.	Alternate AI index: AIQ	1.08	0.34	1850
18.	Weight by 2021 market cap	1.07	0.38	1850
19.	Use SWE employment share	1.23	0.26	1850
20.	Reweight occupations for Revelio coverage	1.38	0.41	1844
21.	Instrument for SWE payroll share with its lagged value	1.24	0.26	1850
22.	Bootstrap SE (block + firm resample)	1.24	0.47	1850

Notes: The Baseline row corresponds to the baseline NAICS3 fixed-effects specification in the main continuous table. Standard errors are heteroskedasticity robust. Row 4 controls for three measures of firms' exposure to generative AI through the product market, as defined in Table VII of Eisfeldt et al. (2026). Row 8 controls for book-to-market, operating profitability, investment (capex to assets), book leverage, Tobin's Q, return on assets, labor intensity, asset tangibility, and market leverage. Row 9 controls for log market capitalization at the end of 2021, and Row 18 weights the regression by the level of market capitalization at the end of 2021. Row 13 estimates firm exposure using daily excess returns from 2022–2025, with daily FF3 factors and daily THNQ excess returns in the first stage. Row 14 estimates firms' AI betas using the unique trading dates corresponding to the day of a major AI model release or one day later. We apply the same trimming procedure as in Table 1 except in Row 19, where we trim at the 99th percentile of the software engineering employment share. Row 20 uses adjusted software engineer payroll shares that reweight occupations so that within each NAICS2 sector, each occupation's employment share in Revelio matches its corresponding employment share in OEWS data. Row 21 instruments the 2021 software engineer payroll share with its 2020 value and reports HC1-style heteroskedasticity-robust standard errors.

the coefficients. The estimates are similar in every year. Finally, in Figure B.3, we show results from estimating the regression separately on each NAICS3 sector with at least 25 firms in our main sample.

3.4 Quantifying the Software Engineering Productivity Gain from AI

This subsection turns to the paper's goal: backing out news about software engineering productivity from the second-stage empirical estimates δ and the model, using the method in Section 2.

The formula. Proposition 1 uses δ to recover the market-implied news about AI-driven software engineering productivity. By equation (10), a movement $\tilde{R}_t^{\text{AI}}$ in the residualized AI index implies a normalized present-value software engineering productivity change of $\kappa \cdot \tilde{R}_t^{\text{AI}}$, where

$$\kappa = \frac{\delta}{\delta^S} = \frac{\Sigma}{\alpha(\varepsilon-1)} \cdot \delta, \qquad \delta^S = \alpha \frac{\varepsilon-1}{\Sigma}, \qquad \Sigma = \sigma + \big[\alpha(\varepsilon-1) - (\sigma-1)\big] \frac{\sigma_\gamma^2}{\gamma(1-\gamma)}.$$

That is, a residualized AI-index return of 1% predicts a normalized present-value software engineering productivity change of κ%. We already have estimates of the regression coefficient δ in hand. We now calibrate the model parameters that determine δ^S.

The demand elasticity ε. The demand elasticity governs the product-market force: how strongly a relative cost advantage translates into relative profit gains. Standard values in the production-network literature are $\varepsilon \approx 4$–6, based on trade-elasticity estimates from Broda and Weinstein (2006). Following Broda and Weinstein (2006), we set our baseline to $\varepsilon = 5$, which corresponds to a markup of $\mu = \varepsilon/(\varepsilon-1) = 1.25$.

The within-firm substitution elasticity σ. The parameter σ governs substitution between software engineering and other labor inputs within the firm. Since software engineers produce software, the most directly relevant evidence is the substitutability between software and other inputs in production.[18] Using Korean firm-level data, Aum and Shin (2024) estimate an elasticity of substitution between software and other labor of 1.6. Our baseline is accordingly $\sigma = 1.6$. This value is also close to the canonical estimate of substitution across education groups: Katz and Murphy (1992) estimate an elasticity of 1.4 between college and non-college labor.

The elasticity σ is the most influential structural parameter: when cross-sectional dispersion in SWE payroll shares (σ_γ^2) is small relative to $\gamma(1-\gamma)$, the aggregate substitution elasticity satisfies $\Sigma \approx \sigma$, so the productivity multiplier κ is approximately proportional to σ. Given its importance, we also explore $\sigma \in \{0.5, 1.0, 1.5, 2.0\}$, a range that deliberately spans the two economically distinct cases: gross complementarity between software engineering labor and non-SWE labor when $\sigma < 1$, and gross substitutability when $\sigma > 1$. The range also reflects the empirical ambiguity in AI's effect on non-SWE labor demand: AI may substitute for exposed tasks and reduce demand for some non-SWE labor, but it may also complement non-SWE workers by shifting effort toward tasks that remain difficult to automate (Acemoglu et al., 2022; Hampole et al., 2025).

Data-based objects. We construct γ, the sales-weighted aggregate software engineering payroll share, directly from the all-sector Compustat-Revelio data, using data for 2021. Unlike in the regression sample, this object retains NAICS 51 and 54, semiconductor-related firms, and constituents of our chosen AI ETFs, because the object here is the sales-weighted aggregate software engineering

[18] Section 4.2 makes this point precise by showing that a richer model with an upstream sector that sells software is isomorphic to our baseline model.

payroll share for the full economy. We still exclude firms in SIC industries 6000-6999 because γ is a sales-weighted object, and sales for financial and real-estate firms are not directly comparable to sales for nonfinancial firms.

We require one additional assumption, which is that the sales-weighted average software engineer payroll share is the same among firms in our sample as in the universe of unobserved firms. With this assumption, we can compute $\gamma = \sum_{j \in \text{observed firms}} \frac{\text{sales}_j}{\text{total observed sales}} \gamma_j$, where γ_j is firm j's software engineer payroll share. We also report results using alternative estimates of γ in Appendix Table A.8.

The above calculation yields $\gamma \approx 0.111$. We compute σ_γ^2, the sales-weighted variance of software engineering payroll shares across firms, in a similar manner by assuming that the sales share-weighted variance of γ_j is the same for observed and unobserved firms. This gives us $\sigma_\gamma^2 \approx 0.0116$. In practice, software engineering payroll shares are small and concentrated: most firms have γ_j close to zero, while a handful of technology-intensive firms have much higher values. This implies $\sigma_\gamma^2 \ll \gamma(1-\gamma)$, so that the between-firm reallocation channel is quantitatively modest.

We construct the labor cost share α using the BEA/BLS KLEMS production accounts, which provide labor compensation and gross output for 61 private-sector industries (excluding federal and state-and-local government). Using this dataset, we obtain the ratio of total labor compensation to total gross output across all private industries, which is 0.297. Because gross output includes profits, we convert this ratio into labor's share of total cost, α, using the markup $\mu = \varepsilon/(\varepsilon - 1)$: $\alpha = \mu \cdot \frac{\text{total labor compensation}}{\text{total gross output}} = 1.25 \times 0.297 \approx 0.372$. Combining $\alpha \approx 0.372$ with $\gamma \approx 0.111$ implies a sales-weighted average cost share of software engineering ≈ 0.041.

As we have discussed, our second-stage regression yields $\delta = 1.24$ (column 4 of Table 1, the specification with NAICS3 fixed effects and size controls).

From November 2022 (the launch of ChatGPT) to December 2025, the cumulative residualized AI-index return (after Fama-French projection) is $\sum_{t=\text{'22 Nov}}^{\text{end '25}} \tilde{R}_t^{\text{AI}} = 22.9\%$. We calculate this cumulative return using factor betas estimated by regressing weekly AI-index excess returns on the Fama-French factors over the same sample period as the first-stage regression (13). Consistent with the definition of $\tilde{R}_t^{\text{AI}}$ in Section 2, we then residualize the AI-index excess return by removing movements associated with the Fama-French factors.[19]

For this calculation, we use the cumulative residualized AI-index return rather than the raw AI-index return. Movements in the risk factors capture discount-rate news that moves all stocks. Therefore, we residualize the AI-index return with respect to these factors before using it to back out software engineering productivity news. Residualization is also required for consistency with

[19] Specifically, we estimate $R_t^{\text{AI}} - R_t^f = a^{\text{AI}} + \sum_i \psi_i^{\text{AI}} F_{it} + \epsilon_t^{\text{AI}}$ and define $\tilde{R}_t^{\text{AI}} = \hat{a}^{\text{AI}} + \hat{\epsilon}_t^{\text{AI}} = R_t^{\text{AI}} - R_t^f - \sum_i \hat{\psi}_i^{\text{AI}} F_{it}$. We retain the AI index's estimated abnormal return when constructing $\tilde{R}_t^{\text{AI}}$ because it reflects the positive AI news during the post-ChatGPT period. Excluding the intercept a^{AI} from the residualized return would exclude this news.

the construction of κ. The multiplier in (8) maps a unit movement in $\tilde{R}_t^{\text{AI}}$ into software engineering productivity news. The second-stage slope δ is estimated after controlling for the risk factors, so it measures firm exposure to the residualized index. Cumulative productivity news is therefore $\kappa \sum_t \tilde{R}_t^{\text{AI}}$, where $\tilde{R}_t^{\text{AI}}$ has been residualized.

Quantifying news about software engineering productivity gains due to AI. We can now calculate the paper's headline estimates of software engineering productivity gains due to AI. Based on the calibrated elasticities $\sigma = 1.6$ and $\varepsilon = 5$, the labor cost share $\alpha = 0.372$, and the data-based objects $\gamma \approx 0.111$ and $\sigma_\gamma^2 \approx 0.0116$, the aggregate substitution elasticity is approximately $\Sigma \approx 1.70$. The corresponding structural gradient is then

$$\delta^S = \alpha \frac{\varepsilon - 1}{\Sigma} = 0.372 \times \frac{4}{1.70} \approx 0.87.$$

Proposition 1 gives the productivity multiplier κ, the percent increase in normalized present-value software engineering productivity implied by a residualized AI-index return of 1%: $\kappa = \delta / \delta^S \approx 1.42$. This means that a residualized AI-index return of 1% implies a 1.42% increase in the normalized present value of software engineering productivity. Aggregating over the event window gives the AI-driven software engineering productivity gain implied by AI news from November 2022 to December 2025:

$$\sum_{t=\text{'22 Nov}}^{\text{end '25}} \mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}_t^{\text{AI}}] = \kappa \cdot \sum_{t=\text{'22 Nov}}^{\text{end '25}} \tilde{R}_t^{\text{AI}} \approx \kappa \times 0.229 \approx 32.6\%, \tag{15}$$

where the cumulative residualized return over the window is 22.9%. If (log) software engineering productivity follows a random walk (permanent innovations), each news term equals the corresponding innovation in (log) productivity. The sum above therefore corresponds to a permanent 32.6% increase in software engineering productivity.

Interestingly, our estimate is similar in magnitude to the task-level gains from AI coding assistants reported in experimental studies: Peng et al. (2023) find a 56% reduction in task completion time for a standardized programming task; Paradis et al. (2025) report a reduction of approximately 21% for a standardized enterprise task at Google; Cui et al. (2026) estimate a 26% increase in pull requests per developer per week, and Gambacorta et al. (2024) find a 55% increase in code output.

Two offsetting forces make our estimate comparable in magnitude to these task-level figures. First, our estimate is forward-looking, reflecting not only today's AI-driven productivity gains but also anticipated future advances in AI models, workflows, and adoption. This forward-looking component pushes our estimate above the gains delivered by current tools alone. Second, because a developer's workflow bundles many complementary tasks, total productivity is constrained by the least-improvable steps, so task-level speed-ups overstate overall gains (Chen and Stratton, 2026; Demirer et al., 2026; Jones and Tonetti, 2026).

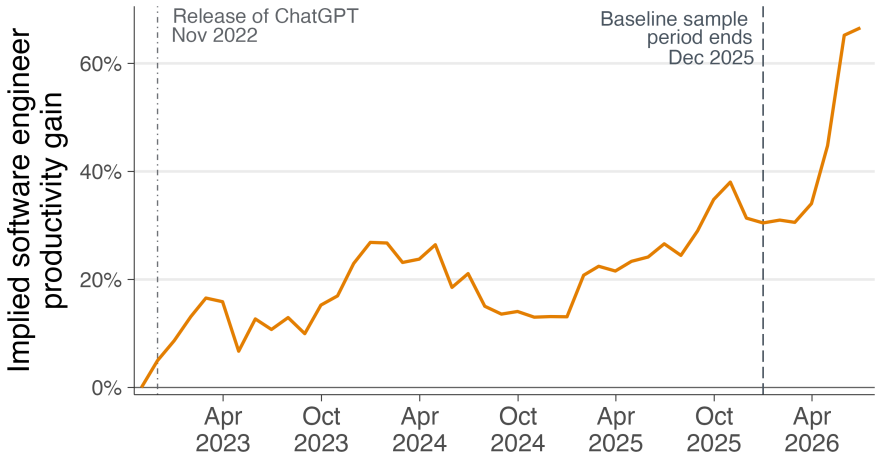
The role of σ. Table A.6 reports the structural gradient δ^S, the productivity multiplier $\kappa = \delta/\delta^S$, and the inferred AI-driven software engineering productivity gain from November 2022 to December 2025 across values of the within-firm substitution elasticity σ between software engineering and non-software-engineering labor.

A higher σ increases the inferred productivity gain. A larger σ lets firms substitute more easily toward software engineering labor when software-engineering productivity rises, raising aggregate substitutability Σ. When Σ is higher, the economy shifts more demand toward software engineering labor after a software engineering productivity gain, bidding up its relative wage and offsetting part of the cost advantage of SWE-intensive firms. This offset lowers the structural gradient δ^S, meaning that a given software engineering productivity gain translates into smaller differences in returns between more and less SWE-intensive firms, i.e., a smaller observed slope δ. Hence a larger underlying software engineering productivity change is needed to explain a given observed slope δ obtained from our two-step empirical approach. On the other hand, a lower σ reduces the inferred productivity gain. The implied productivity gain falls to 22.5% under Cobb-Douglas substitution ($\sigma = 1$) and to 14.0% under complementarity between software engineering and non-software-engineering labor ($\sigma = 0.5$).

Robustness. The AI-driven software engineering productivity gain implied by AI news in (15) depends on the second-stage empirical estimate δ, the structural gradient δ^S, and the cumulative residualized return $\sum_{t=\text{'22 Nov}}^{\text{end '25}} \tilde{R}_t^{\text{AI}}$. Table 2 shows that the second-stage empirical estimate δ is robust across specifications. For the structural gradient δ^S, a potential concern is that we calculate its key inputs, the sales-weighted average software engineer payroll share γ and its cross-sectional variance σ_γ^2, using Revelio firms, which may not be representative. We therefore consider an alternative calibration based on representative data from the Occupational Employment Statistics, which yields $\gamma \approx 0.053$ and $\sigma_\gamma^2 \approx 0.009$. Under this calibration, $\delta^S = 0.847$, and the implied software engineering productivity gain is $\sum_{t=\text{'22 Nov}}^{\text{end '25}} \mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}_t^{\text{AI}}] = 33.6\%$, close to the baseline estimate of 32.6%. Table A.8 further shows that the inferred software engineering productivity gain is robust to alternative calculations of γ.

For the cumulative residualized return $\sum_{t=\text{'22 Nov}}^{\text{end '25}} \tilde{R}_t^{\text{AI}}$, Table A.7 reports robustness to alternative residualization methods. In addition to the baseline FF3 model, we residualize the AI-index returns using a market-only model, an FF5 model, and an FF3 model that replaces the standard market factor with an ex-Magnificent 7 market factor. We also consider a rolling specification in which the factor loadings used to residualize the AI-index returns are estimated over the preceding 104 weeks rather than over the fixed window from January 2022 through December 2025 used in our main analysis. At the lower end, the market-only model yields a somewhat lower estimate because omitting the additional Fama-French factors overestimates the market beta and thus over-subtracts

Figure 2: Software Engineering Productivity News in Real Time



Notes: The figure reports $\kappa \sum_t \tilde{R}_t^{\text{AI}}$ at each month end. Weekly AI-index excess returns are residualized using Fama–French three-factor loadings estimated over the preceding 104 weeks, excluding the current week, and cumulated from November 2022; κ is the baseline productivity multiplier.

the market run-up. At the upper end, the model with the ex-Magnificent 7 market factor yields a somewhat higher estimate because Magnificent 7 firms are important drivers of movements in the standard market factor. We prefer the baseline residualization with three Fama-French factors over those alternatives both because it matches the first-stage specification and because it is standard to assume that the three factors span relevant movements in the SDF.

Real-time Estimates One major benefit of our approach is that it delivers real-time quantification of news about AI-driven software engineering productivity, $\sum_t \mathbb{E}^*[\mathcal{S}_t \mid \tilde{R}_t^{\text{AI}}] = \kappa \cdot \sum_t \tilde{R}_t^{\text{AI}}$. Once the productivity multiplier κ is estimated, we can translate the residualized AI-index return over a given time window into news about AI-driven software engineering productivity over that window.

Figure 2 plots cumulative news about AI-driven software engineering productivity. For each week, we estimate the Fama–French three-factor loadings using the preceding 104 weeks, excluding the current week, and use these loadings to residualize that week's AI-index excess return. We then cumulate the weekly residualized returns from November 2022 onward. We multiply the resulting cumulative return by κ to obtain the software engineering productivity news $\kappa \sum_t \tilde{R}_t^{\text{AI}}$. This series can be updated with each new weekly return and is shown here using the latest available data through the first half of 2026.

While news about software engineering productivity arrived gradually from November 2022 to the end of 2025, the pattern changes from 2026 onward. There is a sharp increase in the first half of 2026, with news about AI-driven software engineering productivity exceeding the total over the previous three years. One likely reason is the widespread introduction and adoption of coding agents, such as Claude Code and Codex, in 2026. We plan to monitor this series as the frontier of AI

continues to expand.

4 The GDP Impact of the Software Engineering Channel

This section extends our baseline model along several dimensions, primarily to translate news about software engineering productivity, due to AI, into its impact on aggregate GDP. Section 4.1 derives this impact using a modified version of Hulten's theorem in the simple environment of Section 2. Section 4.2 introduces an explicit upstream AI sector that supplies an AI input to a software supplier, which bundles a continuum of digital tasks performed by either software engineering labor or AI under comparative advantage. We derive an isomorphism to the baseline model, which leaves the software engineering productivity and GDP impact largely unchanged. Section 4.3 embeds the framework in the full U.S. input-output network. The inferred software engineering productivity gain is nearly unchanged; the GDP impact declines moderately.

4.1 A Modified Version of Hulten's Theorem for the GDP Impact

Having inferred news about AI-driven software engineering productivity $\mathscr{S}_t$ above, we now translate it into its impact on GDP. A software engineering productivity gain lowers firms' marginal costs. One might want to apply the standard version of Hulten's theorem directly to obtain the aggregate GDP effect of this gain using a sales-based Domar weight (each firm's total sales divided by GDP).[20] However, that benchmark holds only in an efficient economy and does not apply here because markups distort production. We instead apply the modified version of Hulten's theorem developed by Baqaee and Farhi (2020) for an economy with distortions, under which the relevant per-firm weight is the cost-based Domar weight $\tilde{\lambda}_j$. In our economy, this weight is the firm's final-expenditure share f_j scaled by the roundabout multiplier $1/\alpha$: $\tilde{\lambda}_j = f_j/\alpha$. The multiplier reflects feedback from the use of final goods as materials inputs: a productivity gain at one firm lowers costs elsewhere and feeds back through the network.

The modified Hulten theorem aggregates firm-level marginal-cost reductions using cost-based Domar weights. A unit increase in software engineering productivity S_t lowers firm j's marginal cost by its software engineering *cost* share $\alpha\gamma_j$. When these cost reductions are weighted by $\tilde{\lambda}_j$, the $1/\alpha$ in the weight cancels the α in the cost share. Integrating across firms, the GDP change induced by the increase in S_t is

$$d\log Y_t\big|_S = \gamma\, d\log S_t, \qquad \gamma = \int_0^1 f_j\gamma_j\, dj,$$

[20] Firm j's sales-Domar weight is $\lambda_{jt} = p_{jt}y_{jt}/C_t$. Because each good's sales equal final consumption plus materials demand, $\lambda_{jt} = \theta f_{jt}$. Thus $\int_0^1 \lambda_{jt}dj = \theta$, where $\theta \equiv 1/[1-(1-\alpha)(\varepsilon-1)/\varepsilon] \geq 1$ is the ratio of gross use of the final-good composite, household consumption plus firms' materials purchases, to GDP.

where γ_j is firm j's software engineering *payroll* share and $\gamma = \int_0^1 f_j \gamma_j \, dj$ its final-expenditure-weighted average.[21] Combining this period-by-period map with news about software engineering productivity from Proposition 1 delivers the headline GDP impact via the software engineering channel.

Proposition 3 (GDP Impact of AI through the Software Engineering Channel)**.** *Define the period-t news in normalized present-value GDP through the software engineering channel as*

$$\mathscr{Y}_t \equiv (1-\beta)\big(\mathbb{E}_t - \mathbb{E}_{t-1}\big)\Big[\sum_{k=0}^{\infty} \beta^k d\log Y_{t+k}\big|_S\Big].$$

In our economy, $\mathscr{Y}_t = \gamma \mathscr{S}_t$. Hence, the best linear predictor of GDP news through the software engineering channel given the contemporaneous movement in the AI index is

$$\mathbb{E}^*[\mathscr{Y}_t \mid \tilde{R}_t^{AI}] = \gamma\kappa \cdot \tilde{R}_t^{AI} = \underbrace{\delta}_{\substack{\text{observed} \\ \text{cross-sectional slope}}} \times \underbrace{\frac{\Sigma}{\alpha(\varepsilon-1)}}_{\substack{\text{inversion:} \\ \text{index return} \to \mathscr{S}}} \times \underbrace{\gamma}_{\substack{\text{cost-based GDP weight:} \\ \mathscr{S} \to \mathscr{Y}}} \cdot \tilde{R}_t^{AI}. \tag{16}$$

Equation (16) provides a transparent sufficient-statistic formula for the GDP impact of AI through the software engineering productivity channel. The brackets decompose the multiplier into three components. The first factor is the observed cross-sectional slope from the second stage of our empirical analysis. The second is the inverse of the structural gradient δ^S, converting the observed slope into the normalized present-value software engineering productivity change per unit movement in the residualized AI index. The third factor translates software engineering productivity into GDP: the final-expenditure-weighted payroll share γ governs how much a given productivity gain raises output.

As an illustrative special case, suppose (log) software engineering productivity follows a random walk. Then each period-t GDP news term is the innovation in (log) GDP, so

$$\mathscr{Y}_t = d\log Y_t\big|_S - \mathbb{E}_{t-1}[d\log Y_t\big|_S] = \gamma\big(d\log S_t - \mathbb{E}_{t-1}[d\log S_t]\big) = \gamma \mathscr{S}_t. \tag{17}$$

We apply Proposition 3 to the productivity gain of 32.6% computed in Section 3.4, implied by the AI news from November 2022 (the launch of ChatGPT) to December 2025. The GDP weight is the final-expenditure-weighted software engineering payroll share $\gamma = \int_0^1 f_j \gamma_j \, dj \approx 0.111$. As discussed above, to study the macroeconomic impact of AI through the software engineering

[21] The resulting GDP elasticity γ exceeds the value the standard sales-Domar Hulten formula would assign to the same cost reductions, $\int_0^1 \lambda_j \alpha \gamma_j \, dj = \alpha\theta\gamma$, by the factor $1/(\alpha\theta)$: the joint markup-and-materials correction, which vanishes only without roundabout production ($\alpha \to 1$) or without markups ($\varepsilon \to \infty$). One might expect from Baqaee and Farhi (2020) an additional reallocation term for the GDP impact, since with markups a productivity gain that shifts resources across firms moves GDP even at first order. It is absent here because all firms have the same labor share and buy the same materials composite, so the markup wedge is uniform across firms and reallocating between them does not change aggregate efficiency (see the proof of Proposition 3 for details). The input-output network of Section 4.3 breaks this symmetry and revives the term, where we find it to be small.

channel, we include firms that produce and sell software (e.g., firms in NAICS 51 and 54) in the calculation of γ because software engineering productivity gains can also lower their marginal costs. The implied normalized present-value GDP change through the software engineering channel is

$$\sum_{t=\text{'22 Nov}}^{\text{end '25}} \mathbb{E}^*[\mathscr{Y}_t \mid \tilde{R}_t^{\text{AI}}] = \gamma \sum_{t=\text{'22 Nov}}^{\text{end '25}} \mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}_t^{\text{AI}}] \approx 0.1110 \times 32.6\% \approx 3.61\%.$$

If (log) software engineering productivity follows a random walk, the GDP news corresponds to a one-time, permanent increase of 3.61% in the level of GDP through the software engineering channel (not an increase in the GDP growth rate of 3.61% per year).[22]

A concern is that the modified Hulten formula in Proposition 3 is an approximation that may work well only for small productivity changes but not for a change of the magnitude we estimate. To address this concern, we also solve the nonlinear model exactly, comparing GDP before and after a 0.326 log-point increase in software engineering productivity. Reassuringly, the exact solution raises the GDP impact only from 0.0361 to 0.0384 log points.[23] The final column of Table A.6 also reports the GDP change implied by the AI-driven software engineering productivity change for different values of σ. The GDP impact inherits the sensitivity of κ to σ noted in Section 3.4. In addition, because the final-expenditure-weighted average software engineering payroll share γ is a key input to the formula in Proposition 3, Table A.8 reports the GDP impact through the software engineering channel using several alternative methods for calculating γ.

As with the software engineering productivity gains, we can track the GDP impact in real time using market prices. We use Proposition 3 to translate the real-time software engineering productivity news in Figure 2 into the corresponding real-time GDP impact shown in Figure B.4. Naturally, there is an acceleration in the GDP impact in the first half of 2026.

We compare our estimate with existing estimates of the macroeconomic effects of AI. Aggregating across all the channels through which AI can affect the economy—software engineering productivity among them—Acemoglu (2025) estimates cumulative TFP level gains of 0.53–0.66% over a decade. On the other hand, Aghion and Bunel (2024) estimate cumulative TFP level gains of 6.8–13% over a decade. To make this comparison, we express our estimate of the GDP impact, via the software channel, in units of aggregate TFP. Following Baqaee and Farhi (2020), changes in aggregate TFP equal changes in GDP net of cost-based-Domar-weighted primary-factor changes. Because the endowments of both primary factors in our environment are fixed, these factor-change terms are zero (see Appendix D.1 for details). Hence, $d\log \text{TFP}_t\big|_S = d\log Y_t\big|_S$. Define normalized

[22] Under the Occupational Employment Statistics-based calibration, $\gamma \approx 0.053$ and the inferred software engineering productivity gain is 33.6%, implying a corresponding GDP increase of approximately $0.053 \times 33.6\% = 1.77\%$.

[23] The exact effect is slightly larger because the productivity gain makes software engineering labor cheaper in effective terms, inducing firms to use more of it and raising the cost-based GDP weight from 0.111 to 0.125. The modified Hulten formula instead holds this weight fixed at its pre-AI value and therefore slightly understates the effect. Appendix H reports the exact calculations for this section and for the production network economy of Section 4.3.

present-value TFP news through the software engineering channel as

$$\mathscr{T}_t \equiv (1-\beta)\big(\mathbb{E}_t - \mathbb{E}_{t-1}\big)\Big[\sum_{k=0}^{\infty} \beta^k d\log \mathrm{TFP}_{t+k}\big|_S\Big].$$

Hence, $\mathscr{T}_t = \mathscr{Y}_t$.

Therefore, our estimated 3.61% normalized present-value GDP gain can also be read as a 3.61% normalized present-value TFP gain. As we have discussed, this number is comparable to a permanent level increase in TFP of 3.61% due to the software engineering channel. Our estimated TFP level gains are therefore higher than the 10-year effect estimated by Acemoglu (2025) and lower than that estimated by Aghion and Bunel (2024).[24] Two points suggest that our 3.61% estimate is a lower bound for the expected effects of AI on TFP. First, we capture the effects of AI on TFP only through the software engineering channel, and other mechanisms likely also contribute. Second, this estimate excludes news after the end of 2025. As we have seen, news in the first half of 2026 approximately doubles the TFP level gains.

4.2 Extension: AI Sector and Task-Based Production

The benchmark model of Section 2 studies AI's impact as a labor-augmenting productivity shock S_t that multiplies software engineering labor. One could argue that AI should instead be modeled as a separately produced commodity, supplied to firms by an upstream sector with its own productivity. One could also argue for modeling AI's economic impact in a task-based framework that features substitution between software engineering labor and AI at the task level, with an endogenous automation margin governing which tasks AI performs, as in Acemoglu and Restrepo (2019, 2022).

This subsection addresses both concerns jointly. We introduce an explicit upstream AI sector that supplies an AI input to a software supplier that bundles a continuum of digital tasks, each performed by either software engineering labor or AI under comparative advantage. The differentiated-good producer, which maps to firms in our baseline regression, then aggregates software inputs, materials, and non-SWE labor through a technology similar to that in Section 2. The software engineering channel operates as in the baseline analysis because of a cost-side isomorphism. Firm marginal cost has the same functional form as the baseline, so AI-driven gains in software engineering productivity reduce marginal cost in the same way. Hence, the procedure to back out the AI-driven software engineering gain and its GDP impact carries through largely unchanged. However, we do have to introduce additional data on firms' software spending.

Environment. Time, preferences, demand, and the outer production nest are exactly as in Section 2. We modify the supply side of the software engineering input in two steps. First, a perfectly

[24] Acemoglu also reports larger effects on GDP than on TFP because his GDP estimates incorporate capital deepening, a channel beyond our simple framework.

competitive upstream AI sector supplies a homogeneous AI input. It is built from compute, a scarce primary factor in fixed supply K that households own and supply inelastically, earning a rental rate $p_{K,t}$. Under a linear, compute-augmenting technology, one unit of compute yields S_t^A units of the AI input, so competition prices the AI input at $p_t^A = p_{K,t}/S_t^A$. A rise in S_t^A lowers the effective price of the AI input for a given compute rental rate.

Second, competitive software suppliers now sell their output to each differentiated-good producer. Supplier j operates a continuum of digital tasks indexed by $z \in [0,1]$. Each task can be performed by software engineering labor or by AI. Let $s_{jt}(z)$ denote the SWE labor allocated to task z for supplier j and $X_{jt}^A(z)$ the corresponding AI input. The task is produced with the linear technology

$$y_{jt}^T(z) = \psi_S(z) s_{jt}(z) + \psi_A(z) X_{jt}^A(z),$$

where $\psi_S(z), \psi_A(z) > 0$ are task-specific factor productivities common across firms. We order tasks so that the comparative advantage of software engineering labor, $\psi_S(z)/\psi_A(z)$, is strictly increasing in z. Higher-z tasks are relatively more productive when performed by SWE labor; lower-z tasks are relatively more productive when performed by AI. The task productivities $\psi_S(z), \psi_A(z)$ and the cross-task elasticity η are uniform across firms.

Task outputs aggregate through a CES with elasticity $\eta > 0$ into a task composite that captures the software output of supplier j,

$$T_{jt} = \Big[\int_0^1 y_{jt}^T(z)^{(\eta-1)/\eta} dz \Big]^{\eta/(\eta-1)}.$$

The software composite T_{jt} enters the differentiated-good producer's technology in the role that effective software-engineering labor $S_t s_{jt}$ plays in Section 2: it combines with non-SWE labor through the CES aggregate

$$H_{jt} = \Big[\phi_j^{1/\sigma} T_{jt}^{(\sigma-1)/\sigma} + (1-\phi_j)^{1/\sigma} L_{jt}^{(\sigma-1)/\sigma} \Big]^{\sigma/(\sigma-1)},$$

which in turn enters the outer Cobb-Douglas nest over H_{jt} and final-good materials, with cost share α on H_{jt}, as in Section 2.

The final-good market clears with output absorbed by consumption and materials, exactly as in Section 2. The SWE and non-SWE labor markets clear, and the compute market clears when the AI sector's compute demand equals the fixed endowment.

Equilibrium task allocation. Each task is performed by whichever factor delivers the lowest effective unit cost. The unit cost of task z is $w_t/\psi_S(z)$ if performed by SWE labor and $p_t^A/\psi_A(z)$ if performed by AI. Monotonicity of $\psi_S(z)/\psi_A(z)$ implies that AI performs all tasks below some threshold $I_t \in (0,1)$ and SWE labor performs all tasks above it, with the threshold pinned down by

the indifference condition

$$\frac{w_t}{\psi_S(I_t)} = \frac{p_t^A}{\psi_A(I_t)}. \tag{18}$$

A rise in the AI sector's productivity S_t^A lowers p_t^A relative to w_t and shifts I_t to the right: AI takes over additional tasks at the margin. In this sense, the task model here captures the notion that AI advances displace software engineering labor.

Splitting the CES task-price integral at the equilibrium threshold I_t shows that the software price p_t^T is a CES aggregator over the two effective input prices w_t and p_t^A with elasticity η,

$$p_t^T = \left[B_A(I_t)(p_t^A)^{1-\eta} + B_S(I_t) w_t^{1-\eta} \right]^{1/(1-\eta)}, \tag{19}$$

where $B_A(I_t) = \int_0^{I_t} \psi_A(z)^{\eta-1} dz$ and $B_S(I_t) = \int_{I_t}^1 \psi_S(z)^{\eta-1} dz$ are the weights on AI and software engineering labor, respectively, and depend endogenously on the automation threshold (see Appendix D.2 for a derivation). Let q_{jt} denote firm j's unit cost of the CES aggregate H_{jt}, and let mc_{jt} denote firm j's marginal cost. Then

$$q_{jt} = \left[\phi_j (p_t^T)^{1-\sigma} + (1-\phi_j) W_t^{1-\sigma} \right]^{1/(1-\sigma)}, \qquad mc_{jt} = \frac{1}{A_t} q_{jt}^{\alpha}. \tag{20}$$

Equation (20) is identical to the benchmark unit cost of Section 2, with the effective software engineering wage w_t/S_t replaced by the software price p_t^T. In this sense, there is a cost-side isomorphism.

Backing out the software engineering productivity gain and its GDP impact. As in Section 2, we seek to infer the AI-driven normalized present-value productivity change from cross-sectional variation in firm AI-index betas. The task-based environment changes the source of the software engineering productivity improvement: it now comes from upstream AI productivity S_t^A. Specifically, holding the SWE wage and the compute rental rate fixed, differentiating (19) gives $d\log p_t^T = -\rho\, d\log S_t^A$, with the threshold response dropping out by the indifference condition (18). The change in the total productivity of the software sector is hence given by $d\log S_t \equiv \rho\, d\log S_t^A$, where $\rho = \frac{B_A(I)(p^A)^{1-\eta}}{(p^T)^{1-\eta}}$ is the AI share of software spending. This share is common across firms because the task productivities and the task elasticity are uniform.

We now show that our procedure for backing out software engineering productivity news due to AI remains effectively unchanged, with γ_j^T in place of γ_j, where $\gamma_j^T \equiv p^T T_j/(p^T T_j + W L_j) = \phi_j (p^T/q_j)^{1-\sigma}$ is the ratio of firm j's total costs for software services to firm j's total costs for the input bundle H_{jt}, the counterpart of the software-engineering payroll share γ_j in the main analysis. Specifically, equation (13), the first stage of our empirical strategy, remains the same, while the second stage now regresses firm betas on γ_j^T. We then invert the resulting estimate δ^T to obtain the normalized present-value AI-driven software engineering productivity gain implied by AI news, as in Proposition 1, replacing γ, σ_γ^2, and Σ with $\gamma^T = \int_0^1 f_j \gamma_j^T dj$, $(\sigma_\gamma^T)^2 = \int_0^1 f_j (\gamma_j^T - \gamma^T)^2 dj$, and

Σ^T, given by the same formula as Σ in Section 3.4 with γ^T and $(\sigma_\gamma^T)^2$ in place of γ and σ_γ^2. The final-expenditure shares $f_j = \omega_j (p_j/P)^{1-\varepsilon}$ are still defined as in Section 2. The GDP impact in Proposition 3 also follows with γ^T replacing γ.

Proposition 4. *In the task-based environment, the best linear predictor of the period-t news in normalized present-value software engineering productivity given the contemporaneous movement in the residualized AI index is*

$$\mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}_t^{AI}] = \kappa \tilde{R}_t^{AI}, \qquad \kappa = \frac{\delta^T}{\delta^{S,T}}, \qquad \delta^{S,T} = \alpha \frac{\varepsilon - 1}{\Sigma^T}, \tag{21}$$

and the corresponding best linear predictor of GDP news through the software engineering channel is

$$\mathbb{E}^*[\mathscr{Y}_t \mid \tilde{R}_t^{AI}] = \gamma^T \kappa \tilde{R}_t^{AI}. \tag{22}$$

Quantifying the productivity gain and its GDP impact. To back out software engineering productivity news in this extension, we need an empirical measure of γ_j^T, defined as firms' spending on the software composite T_{jt} (including both compensation of software engineers and costs of purchasing software services) as a share of total costs of the input bundle H_{jt} (including both spending on the software composite and other labor costs). We construct this measure by augmenting our Revelio data, on software engineer compensation, with data on firms' software budgets from the Ci Technology Database (produced by the marketing company Harte-Hanks). This database has also been used in Bloom et al. (2016) and De Ridder (2024).

The dataset is built from surveys of establishment-level spending on a variety of IT products, most importantly software, and is targeted at technology vendors seeking information on potential customers' IT needs and budgets. We measure γ_j^T as follows:

$$\gamma_j^T = \frac{\text{Software engineer compensation} + \text{Total software budget}}{\text{Total labor compensation} + \text{Total software budget}}. \tag{23}$$

One key concern with this data is measurement error, as the number of establishments surveyed for a particular firm can fluctuate substantially from year to year. Existing work recommends using software budgets for firms' headquarters locations because they are less susceptible to measurement error from inconsistent survey coverage (De Ridder, 2024). So, in the regressions in this section, we instrument for γ_j^T with an alternate version of this variable that replaces the firm's total software budget across all establishments with its software budget for its headquarters location.[25] In Appendix C.4, we provide details on the construction of the software budget measure, and verify that Harte-Hanks's surveys capture a large share of total revenue and software expenditure among private businesses.

[25] Although the first stage of this IV procedure involves two fractions, the relationship is still reasonably linear, as shown by Figure C.2.

We repeat the second-stage specifications reported in Table 1, replacing γ_j with γ_j^T and using its HQ-only counterpart as an instrument. We trim both variables at their respective 99th percentiles within the Harte-Hanks matched sample. The regression coefficient on γ_j^T corresponding to column (4) of Table 1 is $\delta^T = 0.98$. Using γ_j^T calculated based on (23), we obtain $\gamma^T = 0.148$, $(\sigma_\gamma^T)^2 = 0.017$, and $\Sigma^T = 1.721$.[26] With these numbers in hand, Proposition 4 gives $\kappa \approx 1.13$.

Thus a residualized AI-index return of 1% implies a 1.13% increase in the normalized present value of software engineering productivity. Applying the cumulative residualized AI-index return over the window from November 2022 through the end of 2025 gives the implied software engineering productivity gain $\sum_{t='22\,\text{Nov}}^{\text{end }'25} \mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}_t^{\text{AI}}] \approx 25.9\%$. The corresponding GDP news driven by the software channel is $\sum_{t='22\,\text{Nov}}^{\text{end }'25} \mathbb{E}^*[\mathscr{Y}_t \mid \tilde{R}_t^{\text{AI}}] \approx 3.84\%$.

The inferred productivity gain of 25.9% and GDP impact of 3.84% are close to the baseline estimates of 32.6% and 3.61%, respectively. In Figure B.4, we also use the task-based model to translate the real-time cumulative residualized AI-index return into corresponding software engineering productivity news and its implied GDP impact.

4.3 Extension: The Production Network

The model in Section 2 has no input-output structure. This subsection embeds the framework in the full U.S. production network, which consists of 65 BEA Summary sectors, as in Filippucci et al. (2026). These include not only the sectors represented in our empirical analysis but also upstream sectors that produce and sell software, such as those corresponding to NAICS sectors 51 and 54. Each sector is populated by a continuum of firms that differ in software engineering intensity, and each sector buys intermediate inputs from every other sector, while keeping the demand elasticity, markup, and labor-substitution elasticity as in Section 2. This extension allows us to study whether accounting for the production network, in which software is produced upstream and used throughout the economy, affects the inversion used to recover the software engineering productivity gain and the associated GDP impact. Appendix G sets up the economy and provides the results; here we summarize them.

We construct the input-output cost-share matrix from the BEA Make–Use tables. The benchmark sets the across-sector demand elasticity to $\zeta = \varepsilon = 5$, under which the two CES demand tiers collapse to the single CES demand aggregator used in the main analysis. Lowering ζ while maintaining $\zeta \leq \varepsilon$ leaves the headline result essentially unchanged.

[26] We calculate these moments using raw values of γ_j^T because firm-level noise in software budgets is less of a concern for economy-wide averages. Also, applying (23) consistently would require including firms' software budgets in the calibration of the cost share α, since the input bundle now includes purchased software services in addition to labor. This adjustment is immaterial, raising α from 0.372 to roughly 0.388 and lowering κ by about 4%. We therefore use the common value $\alpha = 0.372$ in both the benchmark and task-based environments.

First, the inversion from cross-sectional estimates to an AI-driven software engineering productivity gain remains essentially unchanged. The network effects on intermediate-input prices and sector-level demand are common within sector, so they are absorbed by the second-stage industry fixed effects (Proposition 8). The relative effective software engineering wage response, $d\log w_t - d\log S_t - d\log W_t$ in Section 2, barely moves (-0.573 in the network economy, against -0.587 in the benchmark). The structural gradient δ^S_{in}, which we use to invert the empirical second-stage coefficient δ, also changes only moderately: $\delta^S_{\text{in}} = 0.97$, compared with 0.87 in Section 3.4. The multiplier becomes $\kappa = \delta/\delta^S_{\text{in}} = 1.24/0.97 \approx 1.28$, so the AI-driven software engineering productivity gain implied by AI news over the event window is $\sum_{t=\text{'22 Nov}}^{\text{end '25}} \mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}^{\text{AI}}_t] = \kappa \sum_{t=\text{'22 Nov}}^{\text{end '25}} \tilde{R}^{\text{AI}}_t = \kappa \times 22.9\% \approx 29.3\%$, close to the 32.6% of Section 3.4.

Second, the GDP impact through the software engineering channel is somewhat lowered relative to the benchmark. Proposition 9 extends the cost-based Domar accounting developed in Section 4.1 to the network economy. The GDP elasticity γ^{IO} is a cost-based-Domar-weighted sum of sector software engineering cost shares, minus the small markup-induced factor-reallocation term in Baqaee and Farhi (2020). In our calibration, $\gamma^{\text{IO}} \approx 0.082$. The gap between the network-economy γ^{IO} and the baseline $\gamma \approx 0.111$ mainly reflects the reweighting of sector SWE shares by economy-wide sector sizes implied by the BEA production network and final-demand shares. The baseline is computed from Compustat-Revelio firms, whereas the network calculation includes every sector in the economy. γ^{IO} is smaller because roughly 15 percent of labor income in the model economy is earned in sectors with zero measured software engineering payroll share and that are not in the Compustat-Revelio sample.[27]

Together, the present-value GDP change through the software engineering channel is

$$\sum_{t=\text{'22 Nov}}^{\text{end '25}} \mathbb{E}^*[\mathscr{Y}_t \mid \tilde{R}^{\text{AI}}_t] = \gamma^{\text{IO}}\kappa \cdot \sum_{t=\text{'22 Nov}}^{\text{end '25}} \tilde{R}^{\text{AI}}_t \approx 0.0825 \times 1.277 \times 22.9\% \approx 2.41\%,$$

smaller than the 3.61% benchmark in Section 4.1. Solving the same network economy exactly and nonlinearly raises the implied log GDP change only modestly, from 0.0241 to 0.0255, confirming that the first-order approximation is accurate at the inferred productivity change. In Figure B.4, we also use the production-network model to translate the real-time cumulative residualized AI-index return into corresponding software engineering productivity news and its implied GDP impact.

[27] The corresponding sector-reweighting exercise in the baseline economy similarly lowers the implied GDP impact to 2.44% (row 3 of Table A.8).

5 AI and Innovation

We now extend the framework of Section 2 to allow an AI-driven software engineering productivity gain to affect the macroeconomy through a second channel: innovation. Embedding software engineering labor in a semi-endogenous-growth-style R&D sector adds a new innovation channel: an AI-driven reduction in the cost of effective software engineering labor in R&D facilitates innovation, which in turn raises GDP. This section shows that the procedure in Section 2 used to recover the AI-driven software engineering productivity gain remains unchanged, but that innovation amplifies the GDP impact. Measured in the same units as in our baseline analysis, the implied GDP impact rises from 3.61% to roughly 6.48%. These magnitudes are sizable, both in absolute terms and relative to the other extensions we have considered. How AI affects innovation is therefore key to assessing its GDP impact.

5.1 Environment

Time is discrete, indexed by $t \in \{0,1,\ldots\}$. Household preferences are as in Section 2, and each incumbent variety uses the production technology (1); what changes here is that the set of varieties is endogenous, $j \in [0, N_t]$ with N_t pinned by an endogenous-growth-style R&D sector. We use the same production labor endowments (s, L) and aggregate shocks (S_t, A_t) as in Section 2, drop time variation in the discount factor for clarity, and keep the same numeraire ($P_t = 1$).

Variety producers. With an endogenous continuum of varieties, the household's CES bundle and price index become[28]

$$C_t = \left[\int_0^{N_t} c_{jt}^{(\varepsilon-1)/\varepsilon} dj\right]^{\varepsilon/(\varepsilon-1)} \qquad \text{and} \qquad P_t = \left[\int_0^{N_t} p_{jt}^{1-\varepsilon} dj\right]^{1/(1-\varepsilon)}, \tag{24}$$

with the standard CES demand $c_{jt} = (p_{jt}/P_t)^{-\varepsilon} C_t$, exactly as in Section 2. Materials and the R&D sector final-good input use the same CES composite, so total demand for variety j is $y_{jt} = (p_{jt}/P_t)^{-\varepsilon} Q_t$, where $Q_t \equiv C_t + M_t + X_{R,t}$ is the sum of consumption C_t, aggregate materials M_t, and the R&D final-good input $X_{R,t}$. The final-expenditure-weighted aggregate software engineering payroll share is $\gamma = \int_0^{N^*} f_j \gamma_j dj$, where the SWE payroll share γ_j is as in Section 2, and $f_j = p_j c_j / C = p_j y_j / Q$ are the final-expenditure shares, which integrate to one. Each variety survives to the next period with exogenous probability $\varrho \in (0,1)$, so that $1-\varrho$ is the obsolescence rate. The value of an active variety is therefore

$$V_{jt} = \mathbb{E}_t\left[\sum_{\tau=0}^{\infty} \varrho^\tau \Xi_{t,t+\tau} \pi_{j,t+\tau}\right], \tag{25}$$

[28] In contrast to the fixed weights ω_j integrating to one in Section 2, we deliberately do not normalize the aggregator (24) by the variety mass N_t. This missing normalization is precisely love of variety (Romer, 1990): it is the channel through which a larger variety stock raises real output.

where $\Xi_{t,t+\tau}$ is the SDF from Section 2.

R&D sector. A competitive R&D sector creates new varieties using three inputs: R&D-specific software engineering labor s_R, R&D-specific non-software-engineering labor L_R, and final-good output $X_{R,t}$. The labor endowments for R&D (s_R, L_R) are fixed and earn wages $(w_{R,t}, W_{R,t})$ that equal their marginal value products in R&D. The final-good output $X_{R,t}$ adjusts endogenously. The flow of new varieties follows

$$N_{t+1} = \varrho N_t + \delta_R G(S_t s_R, L_R, X_{R,t})^{\lambda_R} N_t^{\phi_R}, \tag{26}$$

where G is a Cobb-Douglas aggregator with constant exponents α_s, α_L, α_R satisfying $\alpha_s + \alpha_L + \alpha_R = 1$,[29] the software engineering productivity S_t augments software engineering labor in R&D exactly as in production, $\delta_R > 0$ is an R&D productivity parameter, $\lambda_R \in (0, 1]$ governs diminishing returns to R&D inputs, capturing duplication externalities, and $\phi_R < 1$ governs intertemporal knowledge spillovers, with lower values corresponding to stronger fishing-out effects (Jones, 1995; Bloom et al., 2020). The case $\phi_R = 1$ corresponds to fully endogenous growth, while $\phi_R < 1$ corresponds to semi-endogenous growth. Following the empirical scale-effects argument in Jones (1995), we focus on the semi-endogenous case $\phi_R < 1$, in which a permanent increase $d \log S$ in software engineering productivity raises the steady-state level of GDP but does not permanently change the GDP growth rate.

New varieties are drawn from the same type distribution of software engineering intensity as existing varieties, so the expected value of an entrant equals the average incumbent value V^*. Markets clear in the standard way. Appendix D.4 provides details.

5.2 Software Engineering Productivity Gain due to AI, and Its GDP Impact

The inversion for AI-driven SWE productivity gain is unchanged. The inversion in Section 2 maps the observed cross-sectional slope δ from the regression of AI-index betas on software engineering payroll shares into the AI-driven change in software engineering productivity. The same inversion applies in this extension, with the implied present-value productivity change adjusted for obsolescence because financial markets price existing varieties according to their survival probabilities. As in the main analysis, if innovations in log software engineering productivity, $d \log S$, are permanent, the inferred normalized present-value news equals the innovation in $d \log S$ regardless of the obsolescence parameter ϱ.

Appendix D.5 proves that the procedure for backing out software engineering productivity news

[29] Treating software engineering as part of the R&D production function is consistent with evidence that software-intensive firms produce more patents per R&D dollar and that IT complements R&D in patent production (Branstetter et al., 2019).

applies unchanged in the innovation model. The basic intuition is similar to that in Section 2.4: as long as the obsolescence rate (the creative destruction risk) is equal across firms, our approach to inferring software engineering productivity gains from firms' differential exposure to these gains remains valid. We then use the model to quantify the macroeconomic impact of software engineering productivity gains while accounting for innovation. The model helps fill in the missing intercept, namely, the impact of these gains through innovation.

The GDP impact with innovation. Because characterizing transitional dynamics is much more involved and analytically intractable in our environment with semi-endogenous growth, we focus on the first-order response of steady-state GDP $d\log Y^*$ to a permanent change in $d\log S$, holding other aggregate shocks fixed. As discussed above, $d\log Y^*$ is measured in the same units as the normalized present-value GDP news $\mathscr{Y}_t$ studied in the main analysis.

We first decompose the first-order response of steady-state GDP into production-side and innovation effects:

$$d\log Y^* = \gamma\, d\log S + \frac{1}{\alpha(\varepsilon-1)} d\log N^*. \tag{27}$$

The first term is the production-side Hulten effect: cheaper effective software engineering labor lowers unit costs in every production firm, with aggregate impact equal to the final-expenditure-weighted software engineering payroll share γ, the cost-based GDP weight of Proposition 3. The second term is the innovation effect, combining love of variety with the dilution of per-variety output when the fixed production endowments (s, L) are spread across more firms.

Because a positive $d\log S$ also augments software engineering labor in R&D, it raises effective R&D input G, which increases the flow of new varieties and hence the steady-state variety stock. Through the innovation term in (27), the larger variety stock raises GDP. The resulting GDP increase raises the steady-state final-goods input to R&D, X_R^*, which further raises G and generates a feedback loop. The main result characterizes the fixed point of this loop in closed form.

Proposition 5 (Hulten's Theorem with Innovation)**.** *The first-order response of steady-state GDP to a permanent change $d\log S$ is*

$$d\log Y^* = \frac{\gamma + \iota\alpha_s}{1-\iota\alpha_R} d\log S, \tag{28}$$

where $\iota \equiv \lambda_R/[\alpha(\varepsilon-1)(1-\phi_R)]$.[30]

The innovation-augmented GDP coefficient in (28) combines three economically distinct objects: γ, $\iota\alpha_s$, and $\iota\alpha_R$. The first, γ, is the production channel in (27) as in the main analysis: cheaper

[30] The formula requires two regularity conditions: $\iota\alpha_R < 1$ for a stable, positive finite feedback multiplier, and $X_R^*/Y^* = \theta\beta\lambda_R(1-\varrho)\alpha_R/[\varepsilon(1-\beta\varrho)] < 1$ for positive consumption. Here Y^* denotes GDP, equal to consumption plus R&D investment: $Y^* = C^* + X_R^*$, and $\theta = 1/[1-(1-\alpha)\frac{\varepsilon-1}{\varepsilon}]$ is the ratio of gross final-good use Q^* to GDP. Since the steady-state R&D spending share X_R^*/Y^* is constant, consumption has the same first-order log response as GDP: $d\log C^* = d\log Y^*$.

effective software engineering labor lowers unit costs in every production firm, with aggregate impact given by the same weight as in Proposition 3. The second, $\iota\alpha_s$, is the direct innovation channel through which software engineering productivity augments software engineering labor in R&D, thereby raising the effective R&D input G and hence the steady-state variety count by the semi-endogenous factor $\lambda_R/(1-\phi_R)$. The larger variety stock then raises Y^* through love of variety by the additional factor $1/(\alpha(\varepsilon-1))$. The product of these two factors is ι (the semi-endogenous growth "increasing returns" term in the terminology of Jones (2022)), and weighting it by the software engineering share of R&D inputs gives $\iota\alpha_s$. The third term, $\iota\alpha_R$, governs the GDP feedback loop in the denominator: higher GDP raises the final-goods input to R&D, X_R^*, and higher X_R^* raises effective R&D input G, the steady-state variety stock N^*, and hence GDP further.

Quantification. We now quantify the GDP impact of the software engineering channel, taking innovation into account. The exercise inherits $\gamma \approx 0.111$ from Section 4.1, the demand elasticity $\varepsilon = 5$, and the permanent software engineering productivity change $d\log S = 0.326$ from Section 3.4, because Proposition 6 preserves the inversion to AI-driven software engineering productivity gains.

For the new parameters introduced by the innovation block, the innovation-augmented GDP coefficient in (28) depends on (λ_R, ϕ_R) only through the composite $\iota = \lambda_R/[\alpha(\varepsilon-1)(1-\phi_R)]$, so we calibrate this composite directly. In the semi-endogenous accounting of Jones (2022), the corresponding object is the increasing returns parameter $\gamma_J = \lambda_R\sigma_J/\beta_J$, where σ_J maps ideas into output and β_J is the dynamic diminishing-returns parameter in idea production. In our notation, a larger idea (variety) stock raises real value-added output with elasticity $1/(\alpha(\varepsilon-1))$, the variety-margin coefficient in (27), so $\sigma_J = 1/(\alpha(\varepsilon-1))$; the fishing-out term is $\beta_J = 1-\phi_R$. Hence γ_J is exactly our $\iota = \lambda_R/[\alpha(\varepsilon-1)(1-\phi_R)]$. Following Jones (2022), we set $\iota = \gamma_J = 1/3$. This value is close to the corresponding value in Bloom et al. (2020): their baseline aggregate estimate of dynamic diminishing returns is 3.1, measured with ideas in TFP units so that it corresponds to β_J/σ_J here; combined with $\lambda_R = 1$, it implies $\iota = \lambda_R\sigma_J/\beta_J = 1/3.1 \approx 0.323$.

The R&D non-labor share α_R is the cost share of final output used by the R&D sector. Using the NCSES Business Enterprise R&D Survey cost categories and classifying salaries, wages, fringe benefits, stock-based compensation, and temporary staffing as labor-like costs gives $\alpha_s + \alpha_L \approx 0.68$ and hence $\alpha_R \approx 0.32$. Given α_R, the relevant uncertainty is the split of R&D labor between software engineering and other labor. Our baseline sets $\alpha_s = 0.20$ and hence $\alpha_L = 0.48$, based on the Cobb-Douglas patent production function in Kleis et al. (2012), which estimates an IT elasticity close to 0.20.

Under this calibration, the direct innovation channel contributes $\iota\alpha_s \approx 0.0667$, and the GDP-feedback denominator is $1-\iota\alpha_R \approx 0.893$, corresponding to a feedback multiplier of $1/(1-\iota\alpha_R) \approx 1.12$.

Combining these terms gives the innovation-augmented GDP coefficient:

$$\frac{\gamma + \iota\alpha_s}{1 - \iota\alpha_R} = \frac{0.1110 + 0.0667}{0.893} \approx 0.199. \tag{29}$$

Compared to the production-only Hulten coefficient $\gamma \approx 0.111$, the macro impact of AI through the software engineering channel is roughly 1.79 times as large. The amplification decomposes as follows: the direct innovation channel $\iota\alpha_s$ raises the coefficient from 0.111 to 0.178 (a factor of 1.60), and the GDP feedback loop raises it from 0.178 to 0.199 (a further factor of 1.12). Using a permanent shock to software-engineering productivity $d\log S = 0.326$ from Section 3.4, the implied AI-driven GDP change through the software engineering channel is

$$d\log Y^* = \frac{\gamma + \iota\alpha_s}{1 - \iota\alpha_R} \cdot d\log S \approx 0.199 \times 0.326 \approx 6.48\%, \tag{30}$$

compared to the 3.61% production-only baseline of Section 4. In Figure B.4, we also use the innovation model to translate the real-time cumulative residualized AI-index return into the corresponding GDP impact.

Innovation thus substantially amplifies the GDP impact of AI, both in absolute terms and relative to the other extensions we have considered.

6 Conclusion

This paper uses asset prices to infer the macroeconomic impact of AI through the software engineering channel. By exploiting the cross-sectional relationship between firms' AI-index betas and their software engineering payroll shares, and inverting this relationship through a structural model, we estimate that news about AI from November 2022 to December 2025 implies a present-value increase in software engineering productivity equivalent to a permanent increase of roughly 32.6 percent, and a present-value GDP increase through the software engineering channel equivalent to a permanent increase of approximately 3.61 percent. Accounting for the innovation channel can raise the permanent GDP impact to roughly 1.8 times the baseline estimate. A further benefit of our approach is that it quantifies news about AI-driven productivity in real time. Applying our procedure to early 2026 implies large software engineering productivity and GDP gains from AI news during this period. These gains are consistent with the rapid development and adoption of AI coding agents during this period.

Our approach naturally extends to other occupation groups through which AI may affect the macroeconomy by applying the same cross-sectional identification strategy with occupation-specific cost shares. Future work can use this framework to quantify these additional channels and assess their contribution to the macroeconomic impact of AI. Our innovation result suggests a key area for future research: how AI affects the innovation process through software engineering.

References

Acemoglu, Daron. 2025. "The simple macroeconomics of AI." *Economic Policy*, 40(121): 13–58.

Acemoglu, Daron, and Pascual Restrepo. 2018. "The race between man and machine: implications of technology for growth, factor shares, and employment." *American Economic Review*, 108(6): 1488–1542.

Acemoglu, Daron, and Pascual Restrepo. 2019. "Automation and new tasks: how technology displaces and reinstates labor." *Journal of Economic Perspectives*, 33(2): 3–30.

Acemoglu, Daron, and Pascual Restrepo. 2022. "Tasks, automation, and the rise in U.S. wage inequality." *Econometrica*, 90(5): 1973–2016.

Acemoglu, Daron, David Autor, Jonathon Hazell, and Pascual Restrepo. 2022. "Artificial intelligence and jobs: evidence from online vacancies." *Journal of Labor Economics*, 40(S1): S293–S340.

Aghion, Philippe, and Simon Bunel. 2024. "AI and growth: where do we stand?" *Working Paper*.

Aghion, Philippe, Benjamin Jones, and Charles Jones. 2019. "Artificial intelligence and economic growth." In *The Economics of Artificial Intelligence: An Agenda*. 237–290. University of Chicago Press.

Althoff, Lukas, and Hugo Reichardt. 2026. "Task-specific technical change and comparative advantage." *Working Paper*.

Andrews, Isaiah, and Maryam Farboodi. 2025. "Do markets believe in transformative AI?" *NBER Working Paper*.

Aum, Sangmin, and Yongseok Shin. 2024. "Is software eating the world?" *NBER Working Paper*.

Babina, Tania, Alex He, and Renhao Jiang. 2026. "Canaries in the gold mine: Early productivity gains from artificial intelligence creating organization capital." *NBER Working Paper*.

Babina, Tania, Anastassia Fedyk, Alex He, and James Hodson. 2024. "Artificial intelligence, firm growth, and product innovation." *Journal of Financial Economics*, 151: 103745.

Babina, Tania, Anastassia Fedyk, Alex He, and James Hodson. 2025. "Artificial intelligence and firms' systematic risk." *Working Paper*.

Baqaee, David Rezza, and Emmanuel Farhi. 2020. "Productivity and misallocation in general equilibrium." *The Quarterly Journal of Economics*, 135(1): 105–163.

Bertomeu, Jeremy, Yupeng Lin, Yibin Liu, and Zhenghui Ni. 2025. "The economic consequences of disrupted generative AI adoption." *Working Paper*.

Bloom, Nicholas, Charles I. Jones, John Van Reenen, and Michael Webb. 2020. "Are ideas getting harder to find?" *American Economic Review*, 110(4): 1104–1144.

Bloom, Nicholas, Mirko Draca, and John Van Reenen. 2016. "Trade induced technical change? The impact of Chinese imports on innovation, IT and productivity." *The Review of Economic Studies*, 83(1): 87–117.

Bontadini, Filippo, Carol Corrado, Jonathan Haskel, and Cecilia Jona-Lasinio. 2026. "AI as an innovation in the method of innovation: Implications for productivity growth." *AEA Papers and Proceedings*, 116: 36–40.

Borri, Nicola, Aleh Tsyvinski, and Yukun Liu. 2026. "AI premium." *NBER Working Paper*.

Branstetter, Lee, Matej Drev, and Namho Kwon. 2019. "Get with the program: software-driven innovation in traditional manufacturing." *Management Science*, 65(2): 541–558.

Broda, Christian, and David Weinstein. 2006. "Globalization and the gains from variety." *Quarterly Journal of Economics*, 121(2): 541–585.

Brynjolfsson, Erik, Bharat Chandar, and Ruyu Chen. 2026. "Canaries in the coal mine? Six facts about the recent employment effects of artificial intelligence." *Working Paper*.

Brynjolfsson, Erik, Daniel Rock, and Chad Syverson. 2021. "The productivity J-curve: how intangibles complement general purpose technologies." *American Economic Journal: Macroeconomics*, 13(1): 333–372.

Campbell, John, Christopher Polk, and Tuomo Vuolteenaho. 2010. "Growth or glamour? Fundamentals and systematic risk in stock returns." *Review of Financial Studies*, 23(1): 305–344.

Chen, Fiona, and James Stratton. 2026. "Artificial intelligence in the firm." *Working Paper*.

Chow, Trevor, Basil Halperin, and Zachary Mazlish. 2026. "Transformative AI, existential risk, and real interest rates." *Working Paper*.

Cui, Kevin Zheyuan, Mert Demirer, Sonia Jaffe, Leon Musolff, Sida Peng, and Tobias Salz. 2026. "The effects of generative AI on high-skilled work: evidence from three field experiments with software developers." *Management Science*.

Davidson, Tom, Basil Halperin, Thomas Houlden, and Anton Korinek. 2026. "When does automating AI research produce explosive growth? Feedback loops in innovation networks." *NBER Working Paper*.

Demirer, Mert, Leon Musolff, and Liyuan Yang. 2026. "Writing code vs. shipping code: productivity effects across generations of AI coding tools." *NBER Working Paper*.

De Ridder, Maarten. 2024. "Market power and innovation in the intangible economy." *American Economic Review*, 114(1): 199–251.

Eisfeldt, Andrea, Gregor Schubert, Bledi Taska, and Miao Ben Zhang. 2026. "Generative AI and firm values." *Journal of Finance*.

Eloundou, Tyna, Sam Manning, Pamela Mishkin, and Daniel Rock. 2024. "GPTs are GPTs: labor market impact potential of large language models." *Science*, 384(6702): eadj0998.

Fama, Eugene, and James MacBeth. 1973. "Risk, return, and equilibrium: empirical tests." *Journal of Political Economy*, 81(3): 607–636.

Fama, Eugene, and Kenneth French. 1993. "Common risk factors in the returns on stocks and bonds." *Journal of Financial Economics*, 33(1): 3–56.

Fama, Eugene, and Kenneth French. 2015. "A five-factor asset pricing model." *Journal of Financial Economics*, 116(1): 1–22.

Fan, Tianyu. 2025. "The labor market incidence of new technologies." *Working Paper*.

Filippucci, Francesco, Peter Gal, and Matthias Schief. 2026. "Aggregate productivity gains from artificial intelligence: A sectoral perspective." *AEA Papers and Proceedings*, 116: 31–35.

Freund, Lukas, and Lukas Mann. 2026. "Job transformation, specialization, and the labor market effects of AI." *Working Paper*.

Gambacorta, Leonardo, Han Qiu, Shuo Shan, and Daniel Rees. 2024. "Generative AI and labour productivity: a field experiment on coding." *Working Paper*.

Hampole, Menaka, Dimitris Papanikolaou, Lawrence Schmidt, and Bryan Seegmiller. 2025. "Artificial intelligence and the labor market." *NBER Working Paper*.

Hosseini Maasoum, Seyed Mahdi, and Guy Lichtinger. 2026. "Generative AI as seniority-biased technological change: Evidence from U.S. résumé and job posting data." *Working Paper*.

Jones, Charles, and Christopher Tonetti. 2026. "Past automation and future A.I.: how weak links tame the growth explosion." *Working Paper*.

Jones, Charles I. 1995. "R&D-based models of economic growth." *Journal of Political Economy*, 103(4): 759–784.

Jones, Charles I. 2022. "The past and future of economic growth: a semi-endogenous perspective." *Annual Review of Economics*, 14(1): 125–152.

Karger, Ezra, Otto Kuusela, Jason Abaluck, Kevin Bryan, Basil Halperin, Todd Jones, Connacher Murphy, Philip Trammell, Matt Reynolds, Dan Mayland, Ria Viswanathan, Ananaya Mittal, Rebecca Ceppas de Castro, Josh Rosenberg, and Philip Tetlock. 2026. "Forecasting the economic effects of AI." *NBER Working Paper*.

Katz, Lawrence, and Kevin Murphy. 1992. "Changes in relative wages, 1963–1987: supply and demand factors." *Quarterly Journal of Economics*, 107(1): 35–78.

Kleis, Landon, Paul Chwelos, Ronald Ramirez, and Iain Cockburn. 2012. "Information technology and intangible output: the impact of IT investment on innovation productivity." *Information Systems Research*, 23(1): 42–59.

Korinek, Anton, and Donghyun Suh. 2024. "Scenarios for the transition to AGI." *NBER Working Paper*.

Lettau, Martin, and Jessica Wachter. 2007. "Why is long-horizon equity less risky? A duration-based explanation of the value premium." *Journal of Finance*, 62(1): 55–92.

Murphy-Hill, Emerson, Jenna Butler, and Alexandra Savelieva. 2026. "Adoption and impact of command-line AI coding agents: A study of Microsoft's early 2026 rollout of Claude Code and GitHub Copilot CLI." *Working Paper*.

Oberfield, Ezra, and Devesh Raval. 2021. "Micro data and macro technology." *Econometrica*, 89(2): 703–732.

Oster, Emily. 2019. "Unobservable selection and coefficient stability: Theory and evidence." *Journal of Business and Economic Statistics*, 37(2): 187–204.

Paradis, Elise, Kate Grey, Quinn Madison, Daye Nam, Andrew Macvean, Vahid Meimand, Nan Zhang, Ben Ferrari-Church, and Satish Chandra. 2025. "How much does AI impact development speed? An enterprise-based randomized controlled trial."

Peng, Sida, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. 2023. "The impact of AI on developer productivity: evidence from GitHub Copilot." *Working Paper*.

Romer, Paul. 1990. "Endogenous technological change." *Journal of Political Economy*, 98(5): S71–S102.

Trammell, Philip, and Anton Korinek. 2023. "Economic growth under transformative AI." *NBER Working Paper*.

Main proofs

Proof of Lemma 1

The proof has three steps: relative profits given factor prices, the general-equilibrium factor-price response to the software engineering shock, and valuation.

Step 1: relative profits. Because households and firms purchase the same final-good composite, let $M_t \equiv \int_0^1 m_{jt} dj$ denote aggregate materials use and let $Q_t \equiv C_t + M_t$ denote total use of the final-good composite. Since firms charge a constant markup, profits are a constant fraction of revenue. Under CES demand and the numeraire $P_t = 1$,

$$\pi_{jt} = \frac{1}{\varepsilon} p_{jt} y_{jt} = \frac{1}{\varepsilon} \omega_j p_{jt}^{1-\varepsilon} Q_t.$$

Equation (33) in Step 2 implies $Q_t = \theta C_t$ with constant θ. Hence $d\log Q_t = d\log C_t$, and therefore

$$d\log \pi_{jt} = (1-\varepsilon) d\log mc_{jt} + d\log C_t.$$

By Shephard's lemma applied to the unit-cost function, and using that materials are priced in units of the numeraire,

$$d\log mc_{jt} = -d\log A_t + \alpha\gamma_j (d\log w_t - d\log S_t) + \alpha(1-\gamma_j) d\log W_t.$$

Differentiating the CES price index gives the final-expenditure-weighted average cost change,

$$d\log P_t = \int_0^1 f_j d\log p_{jt} dj = -d\log A_t + \alpha\gamma(d\log w_t - d\log S_t) + \alpha(1-\gamma)d\log W_t.$$

The numeraire $P_t = 1$ sets the left side to zero. Subtracting the price-index equation from firm j's unit-cost equation gives the relative unit-cost change

$$d\log mc_{jt} = -\alpha(\gamma_j - \gamma)\big[d\log S_t - (d\log w_t - d\log W_t)\big]. \tag{31}$$

Substituting (31) into the profit equation yields

$$d\log \pi_{jt} = (\varepsilon-1)\alpha(\gamma_j-\gamma)(d\log S_t - (d\log w_t - d\log W_t)) + d\log C_t. \tag{32}$$

Step 2: factor prices. Define

$$x_t \equiv d\log S_t - (d\log w_t - d\log W_t),$$

the change in the effective software engineering cost advantage relative to non-SWE labor. We derive x_t from labor-market clearing.

Define firm j's sales-Domar weight as $\lambda_{jt} \equiv p_{jt}y_{jt}/C_t$. Since $p_{jt}y_{jt} = f_{jt}Q_t$,

$$\lambda_{jt} = \theta f_{jt}, \qquad \theta \equiv \frac{Q_t}{C_t}.$$

Materials are the fraction $1-\alpha$ of each firm's cost, while cost is the fraction $(\varepsilon-1)/\varepsilon$ of revenue. Aggregating across firms therefore gives

$$M_t = (1-\alpha)\frac{\varepsilon-1}{\varepsilon}Q_t.$$

Combining this identity with $Q_t = C_t + M_t$ yields

$$\theta = \frac{Q_t}{C_t} = \frac{1}{1-(1-\alpha)(\varepsilon-1)/\varepsilon}, \tag{33}$$

which is constant. In particular, $\int_0^1 \lambda_{jt}dj = \theta$ and $\int_0^1 \lambda_{jt}\alpha\gamma_{jt}dj = \theta\alpha\gamma_t$, where $\gamma_t = \int_0^1 f_{jt}\gamma_{jt}dj$.

The outer Cobb-Douglas nest between the labor composite and materials fixes labor's share of total cost at α. Since markups are constant, each firm's total cost is the constant fraction $\frac{\varepsilon-1}{\varepsilon}$ of its revenue, and the time-t cost shares $\alpha\gamma_{jt}$ and $\alpha(1-\gamma_{jt})$ of that cost go to SWE and non-SWE labor. Integrating across firms,

$$w_t s = \frac{\varepsilon-1}{\varepsilon}C_t\int_0^1 \lambda_{jt}\alpha\gamma_{jt}dj = \frac{\varepsilon-1}{\varepsilon}C_t\theta\alpha\gamma_t \qquad \text{and} \qquad W_t L = \frac{\varepsilon-1}{\varepsilon}C_t\theta\alpha(1-\gamma_t).$$

Because θ is constant, fixed factor supplies give

$$d\log w_t = d\log C_t + \frac{d\gamma_t}{\gamma} \qquad \text{and} \qquad d\log W_t = d\log C_t - \frac{d\gamma_t}{1-\gamma}.$$

The aggregate demand term $d\log C_t$ cancels in the relative wage:

$$d\log w_t - d\log W_t = \frac{d\gamma_t}{\gamma(1-\gamma)}.$$

It remains to compute $d\gamma_t$. First, CES demand implies

$$d\log f_{jt} = (1-\varepsilon)(d\log p_{jt} - d\log P_t) = (\varepsilon-1)\alpha(\gamma_j-\gamma)x_t,$$

where the second equality uses $d\log p_{jt} = d\log mc_{jt}$, $d\log P_t = 0$, and the relative unit-cost response (31). Second, the CES payroll-share formula implies

$$\log\Big(\frac{\gamma_{jt}}{1-\gamma_{jt}}\Big) = \log\Big(\frac{\phi_j}{1-\phi_j}\Big) + (1-\sigma)(\log w_t - \log S_t - \log W_t),$$

so

$$d\gamma_{jt} = (\sigma-1)\gamma_j(1-\gamma_j)x_t.$$

Therefore, using $df_{jt} = f_j d\log f_{jt}$ to convert the log differential to the absolute differential,

$$d\gamma_t = \int_0^1 \gamma_j df_{jt} dj + \int_0^1 f_j d\gamma_{jt} dj = (\varepsilon-1)\alpha\int_0^1 f_j\gamma_j(\gamma_j-\gamma)dj\cdot x_t + (\sigma-1)\int_0^1 f_j\gamma_j(1-\gamma_j)dj\cdot x_t.$$

Using

$$\int_0^1 f_j\gamma_j(\gamma_j-\gamma)dj = \sigma_\gamma^2 \qquad \text{and} \qquad \int_0^1 f_j\gamma_j(1-\gamma_j)dj = \gamma(1-\gamma)-\sigma_\gamma^2,$$

we obtain

$$d\gamma_t = \Big[(\sigma-1)\gamma(1-\gamma) + \big(\alpha(\varepsilon-1)-(\sigma-1)\big)\sigma_\gamma^2\Big]x_t.$$

Substituting into the relative wage expression gives

$$d\log w_t - d\log W_t = \Big[(\sigma-1) + \big(\alpha(\varepsilon-1)-(\sigma-1)\big)\frac{\sigma_\gamma^2}{\gamma(1-\gamma)}\Big]x_t.$$

Using the definition of x_t,

$$\begin{aligned} x_t &= d\log S_t - (d\log w_t - d\log W_t) \\ &= d\log S_t - \Big[(\sigma-1) + \big(\alpha(\varepsilon-1)-(\sigma-1)\big)\frac{\sigma_\gamma^2}{\gamma(1-\gamma)}\Big]x_t. \end{aligned}$$

Hence

$$x_t = \frac{1}{\Sigma}d\log S_t, \qquad \text{where} \qquad \Sigma = \sigma + \big[\alpha(\varepsilon-1)-(\sigma-1)\big]\frac{\sigma_\gamma^2}{\gamma(1-\gamma)}.$$

Substituting back into (32) yields the period-by-period profit response

$$d\log\pi_{jt} = \delta^S(\gamma_j-\gamma)d\log S_t + d\log C_t, \qquad \text{where} \qquad \delta^S \equiv \alpha\frac{\varepsilon-1}{\Sigma}. \tag{34}$$

Step 3: firm value and software engineering productivity news. The period-by-period profit response (34) applies at every future horizon. Therefore, for each $k \geq 0$,

$$d\log\pi_{j,t+k} = \delta^S(\gamma_j - \gamma)d\log S_{t+k} + d\log C_{t+k}.$$

The Hicks-neutral shock A_t cancels from relative unit costs entirely, and the discount-factor shock β_t does not enter profits directly; the firm-invariant level shift $d\log C_{t+k}$ captures the aggregate demand effect. Log-linearizing firm value (2) to first order gives

$$d\log V_{jt} = (1-\beta)\mathbb{E}_t\Big[\sum_{k=0}^{\infty}\beta^k(d\log\pi_{j,t+k} + d\log\Xi_{t,t+k})\Big].$$

Substituting the horizon-specific profit response and combining the firm-invariant terms yields the firm-value level response

$$d\log V_{jt} = \delta^S\gamma_j(1-\beta)\mathbb{E}_t\Big[\sum_{k=0}^{\infty}\beta^k d\log S_{t+k}\Big] + \mu_t^V, \tag{35}$$

where μ_t^V collects all firm-invariant terms: the present-value aggregate demand effect, the stochastic discount factor effect, and the cross-sectional mean profit response.

The empirical first stage uses returns. To first order, $R_{jt} \approx d\log V_{jt} - d\log V_{j,t-1}$, and since $d\log V_{j,t-1}$ is known at $t-1$, the unexpected return satisfies

$$R_{jt} - \mathbb{E}_{t-1}[R_{jt}] = d\log V_{jt} - \mathbb{E}_{t-1}[d\log V_{jt}].$$

Taking the unexpected component of (35) therefore applies the revision $(\mathbb{E}_t - \mathbb{E}_{t-1})$ to each term. Defining the unexpected firm-invariant component as $\mu_t \equiv \mu_t^V - \mathbb{E}_{t-1}[\mu_t^V]$ gives

$$R_{jt} - \mathbb{E}_{t-1}[R_{jt}] = \delta^S\gamma_j\mathscr{S}_t + \mu_t, \tag{36}$$

where $\mathscr{S}_t$ is the software engineering productivity news defined in (6). Thus, (36) establishes (7). □

Proof of Proposition 1

By (7) in Lemma 1, the unexpected firm return admits the representation

$$R_{jt} - \mathbb{E}_{t-1}[R_{jt}] = \delta^S\gamma_j\mathscr{S}_t + \mu_t,$$

where μ_t is common across firms. Condition (ii) in Section 2.3, $\mathrm{Cov}_t(\mathbb{E}_{t-1}[R_{jt}] - R_t^f, \tilde{R}_t^{\mathrm{AI}}) = 0$, implies $\mathrm{Cov}_t(R_{jt} - R_t^f, \tilde{R}_t^{\mathrm{AI}}) = \mathrm{Cov}_t(R_{jt} - \mathbb{E}_{t-1}[R_{jt}], \tilde{R}_t^{\mathrm{AI}})$. Projecting the excess return $R_{jt} - R_t^f$ on the residualized AI-index return gives

$$\beta_j^{\mathrm{AI}} = \frac{\mathrm{Cov}_t(R_{jt} - R_t^f, \tilde{R}_t^{\mathrm{AI}})}{\mathrm{Var}_t(\tilde{R}_t^{\mathrm{AI}})} = \bar{\beta}^{\mathrm{AI}} + \delta^S\kappa\gamma_j, \tag{37}$$

where

$$\tilde{\beta}^{\mathrm{AI}} \equiv \frac{\mathrm{Cov}_t(\mu_t, \tilde{R}_t^{\mathrm{AI}})}{\mathrm{Var}_t(\tilde{R}_t^{\mathrm{AI}})}$$

is firm-invariant and

$$\kappa = \frac{\mathrm{Cov}_t(\mathscr{S}_t, \tilde{R}_t^{\mathrm{AI}})}{\mathrm{Var}_t(\tilde{R}_t^{\mathrm{AI}})}$$

is the projection coefficient defined in (8). Taking the cross-sectional partial regression coefficient on the payroll share γ_j conditional on controls X_j,

$$\delta = \frac{\mathrm{Cov}_j(\beta_j^{\mathrm{AI}}, \gamma_j \mid X_j)}{\mathrm{Var}_j(\gamma_j \mid X_j)} = \delta^S \kappa,$$

where the second equality follows from (37): β_j^{AI} is affine in γ_j alone, so partialling out X_j does not affect the slope. Rearranging,

$$\kappa = \frac{\delta}{\delta^S} = \frac{\Sigma}{\alpha(\varepsilon - 1)} \cdot \delta.$$

The residualized AI return has unconditional mean zero by condition (i) in Section 2.3. Because $\mathscr{S}_t$ is a conditional-expectation revision, the law of iterated expectations implies $\mathbb{E}[\mathscr{S}_t] = 0$. Therefore, the best linear predictor has no constant term:

$$\mathbb{E}^*[\mathscr{S}_t \mid \tilde{R}_t^{\mathrm{AI}}] = \kappa \cdot \tilde{R}_t^{\mathrm{AI}},$$

which is (10). □

Proof of Proposition 2

We treat the two extensions separately.

Part (i): demand shifter. With state-dependent demand weights, per-period profits satisfy

$$d\log \pi_{jt} = \omega_j' d\log Z_t + (1-\varepsilon) d\log mc_{jt} + d\log C_t,$$

where $\omega_j' = \partial \log \omega_j / \partial \log Z|_{\mathrm{ss}}$. Throughout the derivation through (38), normalize $d\log Z_t = 1$ and hold all other shocks fixed. Thus all differentials below are equilibrium responses per unit of the demand shock. Define $\tilde{w} \equiv d\log w_t - d\log W_t$. Factor-market clearing gives

$$w_t s = \frac{\varepsilon - 1}{\varepsilon} \theta \alpha \gamma_t C_t,$$
$$W_t L = \frac{\varepsilon - 1}{\varepsilon} \theta \alpha (1 - \gamma_t) C_t,$$

where $\theta = Q_t / C_t$ is constant. Since factor supplies are fixed,

$$\tilde{w} = \frac{d\gamma_t}{\gamma(1-\gamma)}.$$

Differentiating $\gamma_t = \int_0^1 f_{jt}\gamma_{jt}dj$, using

$$d\log f_{jt} = \omega'_j + (1-\varepsilon)\alpha(\gamma_j - \gamma)\tilde{w} + c_0$$

and

$$d\gamma_{jt} = (1-\sigma)\gamma_j(1-\gamma_j)\tilde{w},$$

where c_0 is firm-invariant, gives

$$d\gamma_t = \int_0^1 f_j\gamma_j d\log f_{jt}dj + \int_0^1 f_j d\gamma_{jt}dj.$$

Since final-expenditure shares integrate to one, $\int_0^1 f_j d\log f_{jt}dj = 0$, so the first integral is unchanged if γ_j is replaced by $\gamma_j - \gamma$. The firm-invariant term c_0 then drops out because $\int_0^1 f_j(\gamma_j - \gamma)dj = 0$, and

$$\begin{aligned} d\gamma_t = &\int_0^1 f_j(\gamma_j - \gamma)\omega'_j dj \\ &+ \Big[(1-\sigma)\gamma(1-\gamma) + \big(\alpha(1-\varepsilon) - (1-\sigma)\big)\sigma_\gamma^2\Big]\tilde{w}. \end{aligned}$$

Combining these equations with the definition of Σ gives

$$\tilde{w} = \frac{\int_0^1 f_j(\gamma_j - \gamma)\omega'_j dj}{\Sigma\gamma(1-\gamma)}. \tag{38}$$

The firm-specific component of the per-period profit response to a unit demand shock is

$$\omega'_j + (1-\varepsilon)\alpha\gamma_j\tilde{w}.$$

Define the normalized present-value demand-shifter news by

$$\mathcal{Z}_t \equiv (1-\beta)(\mathbb{E}_t - \mathbb{E}_{t-1})\Big[\sum_{k=0}^{\infty}\beta^k d\log Z_{t+k}\Big].$$

The per-period loading is the same at every horizon. Linearizing the value equation (2) and taking the period-t revision therefore adds

$$[\omega'_j + (1-\varepsilon)\alpha\gamma_j\tilde{w}]\mathcal{Z}_t$$

to the unexpected return, up to a firm-invariant term. Because discount-rate exposure remains homogeneous as in the benchmark, the unexpected return is

$$R_{jt} - \mathbb{E}_{t-1}[R_{jt}] = \delta^S\gamma_j\mathcal{S}_t + [\omega'_j + (1-\varepsilon)\alpha\gamma_j\tilde{w}]\mathcal{Z}_t + \mu_t, \tag{39}$$

where μ_t collects all firm-invariant terms.

Condition (ii) in Section 2.3 implies

$$\mathrm{Cov}_t(R_{jt} - R_t^f, \tilde{R}_t^{\mathrm{AI}}) = \mathrm{Cov}_t(R_{jt} - \mathbb{E}_{t-1}[R_{jt}], \tilde{R}_t^{\mathrm{AI}}).$$

Define

$$\kappa^Z \equiv \frac{\mathrm{Cov}_t(\mathcal{Z}_t, \tilde{R}_t^{\mathrm{AI}})}{\mathrm{Var}_t(\tilde{R}_t^{\mathrm{AI}})}.$$

Projecting (39) on the residualized AI-index return gives

$$\beta_j^{\mathrm{AI}} = \tilde{\beta}^{\mathrm{AI}} + \kappa\delta^S\gamma_j + \kappa^Z[\omega_j' + (1-\varepsilon)\alpha\gamma_j\tilde{w}],$$

where $\tilde{\beta}^{\mathrm{AI}}$ is common across firms. Projecting this coefficient on γ_j conditional on X_j gives

$$\delta = \kappa\delta^S + \kappa^Z\delta^Z, \tag{40}$$

where

$$\delta^Z \equiv \frac{\mathrm{Cov}_j(\omega_j', \gamma_j \mid X_j)}{\mathrm{Var}_j(\gamma_j \mid X_j)} + (1-\varepsilon)\alpha\tilde{w}. \tag{41}$$

The first condition of Assumption 1 zeros the covariance term in (41). Its second condition zeros the numerator in (38), so $\tilde{w} = 0$. Hence $\delta^Z = 0$, and (40) reduces to $\delta = \kappa\delta^S$.

Part (ii): discount-rate heterogeneity. With constant demand weights, the return representation (11) is

$$R_{jt} - \mathbb{E}_{t-1}[R_{jt}] = \delta^S\gamma_j\mathscr{S}_t + \eta_j\mathscr{D}_t + \mu_t.$$

Define

$$\kappa^\Xi \equiv \frac{\mathrm{Cov}_t(\mathscr{D}_t, \tilde{R}_t^{\mathrm{AI}})}{\mathrm{Var}_t(\tilde{R}_t^{\mathrm{AI}})}$$

and

$$\delta^\Xi \equiv \frac{\mathrm{Cov}_j(\eta_j, \gamma_j \mid X_j)}{\mathrm{Var}_j(\gamma_j \mid X_j)}. \tag{42}$$

Condition (ii) in Section 2.3 and the first-stage projection give

$$\beta_j^{\mathrm{AI}} = \tilde{\beta}^{\mathrm{AI}} + \kappa\delta^S\gamma_j + \kappa^\Xi\eta_j,$$

where $\tilde{\beta}^{\mathrm{AI}}$ is common across firms. Taking the cross-sectional partial regression coefficient on γ_j conditional on X_j gives

$$\delta = \kappa\delta^S + \kappa^\Xi\delta^\Xi.$$

Under the first condition of Assumption 2, $\kappa^\Xi = 0$. Under its second condition, $\delta^\Xi = 0$. Thus $\kappa^\Xi\delta^\Xi = 0$ under either condition, and $\delta = \kappa\delta^S$.

Both parts therefore imply $\delta = \kappa\delta^S$. Using $\delta^S = \alpha(\varepsilon-1)/\Sigma$ gives

$$\kappa = \frac{\delta}{\delta^S} = \frac{\Sigma}{\alpha(\varepsilon-1)} \cdot \delta.$$

Finally, condition (i) in Section 2.3 gives $\mathbb{E}[\tilde{R}_t^{\mathrm{AI}}] = 0$, while the law of iterated expectations and the definition of $\mathscr{S}_t$ give $\mathbb{E}[\mathscr{S}_t] = 0$. The best linear predictor therefore has no constant term, so the

inversion in Proposition 1 continues to hold. □

Proof of Proposition 3

The proof has four steps.

Step 1: state the Baqaee–Farhi decomposition. All shares and cost-based weights without time subscripts are evaluated at the no-shock steady state around which we take the first-order approximation. We use Theorem 1, equation (4), of Baqaee and Farhi (2020), together with the fictitious-producer construction in their Section II.A. Introduce a competitive producer S that transforms raw SWE labor into effective SWE services with productivity S_t. Its unit price is $p_t^S = w_t / S_t$, and its markup is one. The markups of downstream firms are also fixed. Baqaee and Farhi's equation (4) therefore becomes

$$d\log Y_t\big|_S = \tilde{\lambda}_S d\log S_t - \tilde{\Lambda}_s d\log \Lambda_{s,t} - \tilde{\Lambda}_L d\log \Lambda_{L,t}, \tag{43}$$

where $\tilde{\lambda}_S$ is the cost-based Domar weight of the effective-SWE producer, $\tilde{\Lambda}_s$ and $\tilde{\Lambda}_L$ are the cost-based weights of the two raw factors, and

$$\Lambda_{s,t} = \frac{w_t s}{Y_t}, \qquad \Lambda_{L,t} = \frac{W_t L}{Y_t}.$$

Step 2: compute the cost-based weights. Every downstream firm spends the share $1-\alpha$ of its cost on the same materials composite, whose final-expenditure share on good j is f_j. The cost-based input-output coefficient from buyer k to supplier j is therefore

$$\tilde{\Omega}_{kj} = (1-\alpha) f_j.$$

The downstream cost-based Domar weights satisfy

$$\tilde{\lambda}_j = f_j + \int_0^1 \tilde{\lambda}_k \tilde{\Omega}_{kj}\, dk = f_j \sum_{r=0}^{\infty} (1-\alpha)^r = \frac{f_j}{\alpha}.$$

The effective-SWE producer supplies the share $\alpha\gamma_j$ of firm j's cost. Since producer S spends all its cost on raw SWE labor, its cost-based Domar weight and the raw-factor weights are

$$\begin{aligned} \tilde{\lambda}_S &= \int_0^1 \tilde{\lambda}_j \alpha\gamma_j\, dj = \int_0^1 f_j \gamma_j\, dj = \gamma, \\ \tilde{\Lambda}_s &= \gamma, \\ \tilde{\Lambda}_L &= \int_0^1 \tilde{\lambda}_j \alpha(1-\gamma_j)\, dj = 1-\gamma. \end{aligned} \tag{44}$$

Step 3: show that the reallocation term vanishes. Let downstream gross sales be θ times GDP. Aggregate materials spending is $(1-\alpha)/\mu$ times downstream gross sales, so

$$\theta = \frac{1}{1-(1-\alpha)/\mu}.$$

Downstream total cost is gross sales divided by μ. Since households and materials buyers both spend according to f_{jt}, firm sales remain proportional to f_{jt}, while each firm pays the share $\alpha\gamma_{jt}$ of cost to effective SWE services. Hence, at every date,

$$\Lambda_{s,t} = \frac{\alpha\theta}{\mu}\gamma_t, \qquad \Lambda_{L,t} = \frac{\alpha\theta}{\mu}(1-\gamma_t).$$

The common multiplier $\alpha\theta/\mu$ is constant. Substituting the cost-based weights in (44) and these factor-income shares into the reallocation term in (43) gives

$$\begin{aligned}\tilde{\Lambda}_s d\log\Lambda_{s,t} + \tilde{\Lambda}_L d\log\Lambda_{L,t} &= \gamma d\log\gamma_t + (1-\gamma)d\log(1-\gamma_t)\\ &= d\gamma_t - d\gamma_t\\ &= 0.\end{aligned}$$

Thus the factor-reallocation term vanishes even though the individual factor-income shares may move. This cancellation uses the common markup, common outer labor share, and common materials composite. Substituting this result and $\tilde{\lambda}_S = \gamma$ from (44) into (43) yields

$$d\log Y_{t+k}\big|_S = \gamma d\log S_{t+k}. \tag{45}$$

Step 4: pass from flow effects to news. Applying the normalized present-value news operator to (45) period by period gives

$$\begin{aligned}\mathscr{Y}_t &= (1-\beta)(\mathbb{E}_t - \mathbb{E}_{t-1})\Big[\sum_{k=0}^{\infty}\beta^k d\log Y_{t+k}\big|_S\Big]\\ &= \gamma\mathscr{S}_t.\end{aligned}$$

Combining this relation with (10) and using linearity of the best linear predictor gives

$$\mathbb{E}^*[\mathscr{Y}_t \mid \tilde{R}_t^{\mathrm{AI}}] = \gamma\kappa\tilde{R}_t^{\mathrm{AI}}, \qquad \gamma\kappa = \frac{\gamma\Sigma}{\alpha(\varepsilon-1)}\delta,$$

which proves (16). □