Software-Specific Technical Change

Xian Jiang
UC Davis

Hannah Rubinton
FRB of St. Louis

November 3, 2026

Preliminary and incomplete.

Abstract

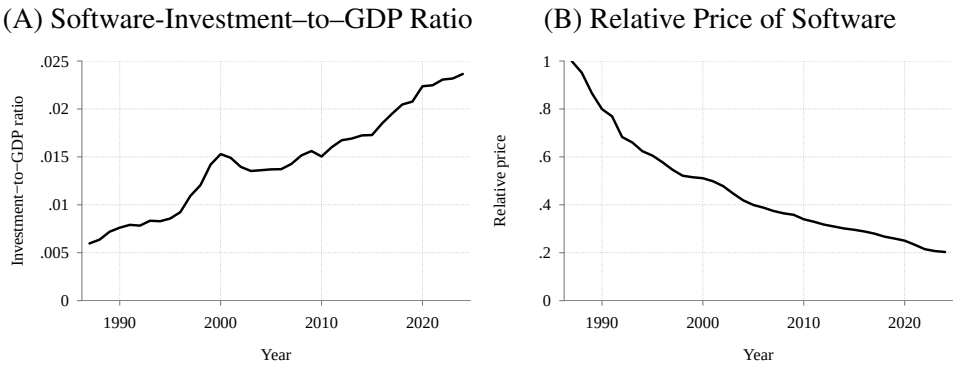
This paper extends the neoclassical growth model to incorporate non-rival software capital in order to study the aggregate implications of software-specific technical change. Over the past decades, software prices have fallen sharply while software has become an increasingly important factor of production. Unlike traditional capital, software is non-rival: a given software stock can be used by many firms at once. Consequently, the flow of software services used in final-good production can exceed the software capital produced by the software providers. Relative to a baseline in which software is modeled like traditional rival capital, a software productivity shock has a larger effect on output. The strength of this amplification increases with the importance of software in production and the response of the software user base, but is attenuated by software specificity. Our main growth accounting exercise implies that software-specific technical change and the scale effects from non-rival software capital account for 21% of growth in output per worker between 1997 and 2024 when software is fully shareable. Modeling software as rival would miss nearly half of the contribution from these channels. Finally, we use the model to discuss the measurement of software capital in the national accounts.

1 Introduction

Software production has become dramatically more efficient over the past three decades. As a result, the relative price of software fell by 80 percent, while software investment rose to 2.4 percent of GDP by 2024 (Figure 1). These improvements are analogous to investment-specific technical change in traditional capital (Greenwood et al., 1997), but software differs in a fundamental respect: it is non-rival. Once produced, the same piece of software can be used by many firms simultaneously. This distinction is increasingly salient in light of recent advances in artificial intelligence (AI), where a common underlying model can be deployed across many users. Motivated by this distinction, we ask how non-rivalry alters the aggregate impact of software-specific technical change.

To answer this question, we extend the neoclassical growth model to incorporate non-rival software capital. As in the standard investment-specific technical change mechanism, an increase in software productivity lowers the cost of producing software capital and therefore the rental price of software services, increasing software use and adoption. Non-rivalry creates a feedback through the size of the user base. As more firms adopt software, the user cost of the underlying software stock is spread across more users, further lowering the rental price and inducing additional adoption and software use. Through this feedback loop, non-rivalry amplifies the aggregate effects of software-specific technical change.

Figure 1: U.S. software investment share and price index



Notes: Panel (A) plots private fixed investment in software divided by nominal GDP from 1987 to 2024. Panel (B) plots the price of software relative to consumption, normalized to one in 1987.

To fix ideas, consider Microsoft Excel. Once a version is developed, it can be sold simultaneously to many users. In contrast, traditional capital is rival: if 10 firms want a robot arm, then 10 robot arms must be produced.[1] The benefits of non-rivalry, however, are limited by the *speci-*

[1]This distinguishes the non-rivalry of software from the non-rivalry of ideas. A robot-arm producer can reuse its

ficity of software. If software is highly tailored to one firm's needs, then the fact that it can be used simultaneously by another firm matters less. For example, if one firm uses Excel primarily for accounting and another for regression analysis, Microsoft must support different features for different users (e.g., pivot tables, macros, and other tools). Meeting more heterogeneous needs requires a larger and more complex software product than if all users had perfectly aligned needs.

Section 2 shows a simple extension of the neoclassical growth model in which software capital is non-rival but subject to specificity. A continuum of monopolistically competitive software producers accumulate software capital and rent it to final-good firms. Software is non-rival but excludable: each producer holds exclusive rights to its variety and charges firms for its use, even though the same software can serve multiple firms simultaneously. The extent of this reuse is limited by the specificity of firms' software needs. We capture this feature by requiring each software producer to maintain a stock at least as large as a CES-type aggregator of the software demanded by each of the endogenous mass of M final-good firms.

The key mechanism is the distinction between the stock of software capital produced by software firms and the flow of software services used by final-good firms in production. In equilibrium, software services exceed the underlying stock by a factor of M^{ρ}, where ρ governs the degree of specificity (we sometimes refer to the opposite of specificity as shareability). This generates aggregate scale effects. When an increase in the labor force expands the number of software users, a given software stock can be reused more broadly, causing output per worker to rise.

Non-rivalry amplifies the impact of software-specific technical change. The strength of the amplification depends on the degree of specificity and on how strongly the number of users, M, responds to the productivity shock. When software is highly specific, it behaves like traditional rival capital, and the effect of the shock is governed by software's share in production. When software is less specific, the expansion of the user base allows the same software capital to provide more services, increasing the effect on output relative to a model that treats software as rival.

We then quantify the importance of software-specific technical change for U.S. growth between 1997 and 2024. Specifically, we decompose growth in output per worker into contributions from software-specific technical change, investment-specific technical change, labor-force growth, and residual Hicks-neutral productivity growth. Section 4 extends the model for the quantitative analysis and describes its calibration, while Section 5 presents the main results. Together, software-specific technical change and the scale effects generated by non-rival software capital account for 21% of growth between 1997 and 2024 when software is fully shareable. The rival benchmark misses almost half of this contribution.

ideas in each unit it produces, but it still must physically produce as many robot arms as it intends to sell.

Finally, Section 6 discusses the measurement of software in the national accounts. Software non-rivalry creates two distinct economic objects: the software capital stock produced by software producers and the software services used by final-good firms. For traditional rival capital, these two objects coincide. With non-rival software, however, the same stock can serve multiple users, allowing software services to exceed the underlying capital stock. We describe the BEA's current methodology for measuring software capital and show how applying this methodology affects the growth accounting in our model.

Related Literature. Our paper relates first to the literature on investment-specific technical change. Greenwood et al. (1997) and Fisher (2006) show how improvements in the production of capital goods, reflected in declining relative prices, contribute to capital deepening and economic growth. We study the analogous force for software, but depart from the standard framework by allowing software capital to be non-rival. Non-rivalry introduces a reuse margin: improvements in software-producing technology expand adoption, which in turn allows a given underlying software stock to generate services for a larger set of users. Our emphasis on adoption and diffusion also connects to work on information technology and general-purpose technologies (Greenwood and Jovanovic, 1999; Jovanovic and Rousseau, 2005).

Second, our paper relates to the literature on intangible capital and productivity measurement (Corrado et al., 2009; McGrattan, 2020). Relatedly, Crouzet et al. (2023) study the implications of partial non-rivalry in intangible capital for aggregate growth. We focus specifically on software and distinguish between the underlying software stock and the flow of services generated through reuse. This distinction matters for growth accounting because gains from reuse can appear as either capital deepening or measured TFP, depending on how software is measured. Our analysis also speaks to recent debates over the productivity effects of AI. Like software more generally, an underlying AI model is at least partly non-rival, so its aggregate effects may depend not only on improvements in the technology itself but also on the scale of its adoption and reuse (Brynjolfsson et al., 2021).

2 The Neoclassical Growth Model with Non-Rival Software

In this section, we develop a model to study the implications of software-specific technical change. The model stays close to the neoclassical growth model, but considers software as a distinct form of capital that is non-rival.

Because of the non-rivalry, the same software capital stock can be used by many final-good firms at the same time. The economy's ability to benefit from this non-rivalry is limited by the

specificity of firms' software needs. If each firm requires a different type of software, the gains from sharing are muted.

2.1 Model Environment

There is a unit mass of monopolistically competitive software producers. They accumulate software capital over time and rent it to final-good firms.

There is a mass M_t of perfectly competitive identical final-good firms that produce a homogeneous final good. Because software is non-rival, a given stock can be deployed across these firms, so M_t plays a central role in shaping the aggregate effects of non-rivalry. M_t is determined endogenously through free entry.

A representative household supplies labor L_t. It owns the physical capital stock and the software and final-good firms, and receives their profits. Each period, it chooses investment in physical capital. Time is discrete and infinite.

2.2 Final Good Firms

Final goods are produced by an endogenous mass M_t of identical perfectly competitive firms. Operating a final-good firm requires a fixed cost (f_c) each period, paid in units of the final good. Conditional on operation, a final-good producer (i) combines labor (l_{it}), rival traditional capital (k_{it}), and non-rival software (s_{it}) according to the decreasing-returns-to-scale technology, given by

$$y_{it} = z_t \left(l_{it}^{\alpha_l} k_{it}^{\alpha_k} s_{it}^{\alpha_s} \right)^{\eta}, \qquad \text{where} \qquad \alpha_k + \alpha_s + \alpha_l = 1, \quad \eta < 1. \tag{1}$$

The software input is in turn a CES aggregate of a continuum of software varieties indexed by $j \in [0,1]$:

$$s_{it} = \left(\int_0^1 s_{ijt}^{\frac{\varepsilon-1}{\varepsilon}} \, dj \right)^{\frac{\varepsilon}{\varepsilon-1}}, \tag{2}$$

where $\varepsilon > 1$ is the elasticity of substitution across software varieties.

Taking the wage (w_t), the rental rate of traditional capital (r_t^k), and the rental rates of software varieties ($\{r_{jt}^s\}_{j\in[0,1]}$) as given, profit maximization of final good firms implies the following

demand for software variety j:

$$s_{ijt} = \eta \alpha_s y_{it} (R_t^s)^{\varepsilon-1} (r_{jt}^s)^{-\varepsilon}, \qquad \text{where} \qquad R_t^s = \left(\int_0^1 \left(r_{jt}^s\right)^{1-\varepsilon} dj \right)^{\frac{1}{1-\varepsilon}} \tag{3}$$

is the rental-rate index for the software bundle. Throughout, we use the final good as the numeraire and normalize its price to one.

2.3 Software Producers

Software is produced by a unit mass of monopolistically competitive software producers. Each software firm produces a distinct variety of software, indexed by j. In each period, software producer (j) creates new software by converting final goods I_{jt}^s to software with software-specific productivity q_t^s. We assume there is time-to-build, so the newly created software can only be used in the next period. Software producers accumulate their software stock according to

$$s_{j,t+1} = (1-\delta_s) s_{jt} + q_t^s I_{jt}^s, \tag{4}$$

where δ_s is the depreciation rate of software to software producers.

In each period, software producers rent their capital stock s_{jt} to all final-good firms with a rental rate r_{jt}^s, and collect revenue $\int_0^{M_t} r_{jt}^s s_{ijt} di$. With symmetric final-good firms, their revenue can be written as $M_t r_{jt}^s s_{ijt}$.

Software is non-rival so the same unit of software capital can serve multiple final-good firms simultaneously. The extent to which software producers can supply the same capital to many firms, however, depends on the *specificity* of firms' software needs. If each final-good firm has entirely specific requirements, the software producer must create different code for each firm, limiting the gains from non-rivalry. By contrast, when firms' needs overlap, the same code can serve multiple firms.

The key deviation from previous literature is the *resource constraint of software*, which captures both the non-rivalry and the specificity of software:

$$s_{jt} \geq \left(\int_0^{M_t} s_{ijt}^{\frac{1}{1-\rho}} di \right)^{1-\rho} = s_{ijt} M_t^{1-\rho}, \tag{5}$$

where the second equality follows from the assumption that final-good firms are identical. The resource constraint says that in order for the software producer to satisfy the demand of the final-good firms, the software stock they produce, s_j, must be greater than a CES-like aggregate of the

software that each firm demands of variety j, s_{ij}.

The parameter ρ governs the degree of specificity. As $\rho \to 1$, software becomes fully shareable: the same unit can serve every final-good firm. The producer therefore needs to supply only enough software to meet the largest firm's demand, so $s_{jt} = \max_i\{s_{ijt}\}$. When $\rho = 0$, by contrast, software is fully firm-specific and effectively behaves like rival traditional capital. The producer must supply a separate unit to each firm, implying $s_{jt} = s_{ijt}M_t$ when final-good firms are identical. Software remains technologically non-rival, but full specificity eliminates any effective reuse across firms.

The software producer's problem is dynamic. Each chooses its software investment (I^s_{jt}) and the rental rate of software (r^s_{jt}) in each period to maximize its discounted stream of profits:

$$\max_{\{I^s_{jt}, s_{j,t+1}, r^s_{jt}\}} \sum_{t=0}^{\infty} \Lambda_t \left(M_t r^s_{jt} s_{ijt} - I^s_{jt} \right), \tag{6}$$

subject to the software demand in Equation (A.3), software accumulation in Equation (4), and the resource constraint in Equation (5). Because software firms are owned by households, Λ_t is the household discount factor implied by their Euler equation that we specify later.

The marginal costs of software. In this model, there are three distinct notions of marginal cost, which help clarify how software non-rivalry operates. The first is the user cost of expanding the software stock. Let κ_j denote the steady-state user cost of one additional unit of software capital, s_j. Because producing an additional unit of $s_{j,t+1}$ costs $1/q^s$ units of the final good, the user cost is

$$\kappa_j = \underbrace{\frac{1}{q^s}}_{\substack{\text{marginal cost of producing} \\ \text{one additional unit of } s_{j,t+1}}} \left(\frac{1}{\beta} - 1 + \delta_s \right).$$

The second is the intensive-margin marginal cost of increasing the software services provided to all existing customers, s_{ij}:

$$MC^{\text{int}}_{ij} = \frac{\kappa_j}{M^\rho}. \tag{7}$$

The intensive margin marginal cost is equal to the user cost of an additional unit of software divided by M^ρ. In other words, the user cost of the software stock is shared across customers, with the extent of cost sharing governed by ρ. When $\rho = 0$, software is fully specific, so the cost cannot be shared and $MC^{\text{int}}_{ij} = \kappa_j$. When $\rho = 1$, the user cost is shared equally by the M final-good firms.

The third is the extensive-margin marginal cost of supplying a unit of software to an additional firm:

$$MC^{\text{ext}}_{ij} = (1-\rho) MC^{\text{int}}_{ij}.$$

When $\rho = 1$, software is fully shareable and the marginal cost of serving an additional customer is zero: the existing code can simply be supplied to the new user. When $\rho < 1$, serving a new customer requires some additional software capital because the needs of the new firm do not overlap completely with those of existing customers.

When $\rho = 0$, all three marginal costs coincide. Accumulating another unit of software capital, providing more software to an existing customer, and serving an additional customer all have the same marginal cost.

The software rental rate. Each software producer takes the number of users, aggregate output, and the aggregate software rental-rate as given because its individual pricing decision does not affect aggregate variables. Its pricing decision therefore operates only along the intensive margin and does not internalize the extensive-margin response of the user base. Standard CES pricing then implies that, in steady state, producer j charges a markup over its intensive-margin marginal cost:

$$r_j^s = \frac{\varepsilon}{\varepsilon - 1} MC_{ij}^{\text{int}}. \tag{8}$$

Within each period, the producer takes its existing software stock as given. Because $\varepsilon > 1$, revenue is decreasing in the rental rate along the demand curve. The producer therefore lowers its rental rate until demand exhausts its available software capacity. Intertemporally, it chooses how much software capital to accumulate so that the Euler equation holds:

$$\frac{1}{q_t^s} = \frac{\Lambda_{t+1}}{\Lambda_t} \left[\frac{\varepsilon - 1}{\varepsilon} M_{t+1}^{\rho} r_{j,t+1}^s + \frac{1 - \delta_s}{q_{t+1}^s} \right]. \tag{9}$$

The left-hand side is the current marginal cost of producing one additional unit of next-period software capital. The right-hand side is its discounted marginal benefit, consisting of the marginal rental revenue generated next period and the continuation value of the undepreciated software capital.

In steady state, combining Equation (9) with final-good firms' demand for software and the binding capacity constraint yields the software stock

$$s_j = \frac{\varepsilon - 1}{\varepsilon} \frac{\eta \, \alpha_s y_i M}{\frac{1}{q^s} \left(\frac{1}{\beta} - 1 + \delta_s \right)}.$$

Because aggregate output is $Y = y_i M$, the software stock is proportional to aggregate output. The steady-state accumulation equation, Equation (4), then implies that software investment by pro-

ducer j is

$$I_j^s = \frac{\delta_s}{q^s} s_j. \tag{10}$$

2.4 The Household Problem

A representative household has preferences

$$\sum_{t=0}^{\infty} \beta^t \log C_t, \tag{11}$$

where C_t denotes consumption and $\beta \in (0,1)$ is the discount factor. The household supplies labor L_t inelastically and owns the physical capital stock, the final-good firms, and the software producers. It receives wage income, rental income from physical capital, and aggregate profits from software producers, Π_t^s. Final-good firms earn zero profits. Each period, the household converts final goods (I_t^k) into new physical capital with investment-specific productivity q_t^k. The physical capital stock evolves according to

$$K_{t+1} = (1-\delta_k)K_t + q_t^k I_t^k, \tag{12}$$

where δ_k is the depreciation rate of physical capital.

The household chooses consumption (C_t) and physical investment (I_t^k) to maximize its preference, subject to the budget constraint:

$$C_t + I_t^k = w_t L_t + r_t^k K_t + \Pi_t^s, \tag{13}$$

and capital accumulation equation (12). The associated Euler equation is

$$\frac{1}{q_t^k} = \beta \frac{C_t}{C_{t+1}} \left(r_{t+1}^k + \frac{1-\delta_k}{q_{t+1}^k} \right). \tag{14}$$

In steady state, with $q_t^k = q^k$ and $C_{t+1} = C_t$, this implies the standard rental rate for physical capital:

$$r^k = \frac{1}{q^k} \left(\frac{1}{\beta} - 1 + \delta_k \right). \tag{15}$$

2.5 General Equilibrium

Free entry. The mass of final-good firms M_t is pinned down by free entry. Producing requires paying a per-period fixed cost f_c, and there is an unbounded mass of potential entrants. Entry continues until the free-entry condition holds:

$$\pi_{it} = (1-\eta)y_{it} = f_c. \tag{16}$$

Because varieties are symmetric and we normalize their measure to one, the price index collapses to the common rental rate, $R_t^s = r_{j,t}^s$ for all j. Likewise, each firm chooses the same quantity of each variety, so the CES aggregate satisfies $s_{i,t} = s_{ijt}$ for all j. In other words, the total software input $s_{i,t}$ equals the per-variety quantity rented by firm i, and the bundle price index equals the price of an individual variety.

Given the population L, fixed costs f_c, production parameters $\{\eta, \alpha_s, \alpha_k, \alpha_l, z\}$, elasticity of substitution across software varieties ε, software production parameters $\{q^s\}$, depreciation rates $\{\delta_s, \delta_k\}$, investment productivity q^k, discount factor β, and software specificity ρ, a steady-state general equilibrium is a set of prices $\{r^k, R^s, w\}$ such that:

- The mass of final-good firms M satisfies the free-entry condition in Eq. (16).
- Software producers optimize; software investment satisfies Eq. (10) and the software rental rate satisfies Eq. (8).
- Final-good firms choose inputs optimally according to Eq. (1).
- Households optimize, and the physical-capital rental rate satisfies Eq. (15).
- Labor, software, capital, and final-good markets clear.

3 The Implications of Non-Rival Software Capital

3.1 Software Capital Versus Software Services

In this economy, the non-rivalry and shareability of software generates aggregate increasing returns to scale. Consider the stock of software capital. Aggregate software capital evolves according to

$$S_t = (1-\delta_s)S_{t-1} + q_{t-1}^s I_{t-1}^s.$$

We drop the j subscript because the mass of software firms is normalized to one, so $S_t = \int_0^1 s_{jt}\, dj = s_{jt}$.

By contrast, let X_t denote total software services, the effective software input summed across final-good firms. X_t is given by

$$\begin{aligned} X_t &= s_{it} M_t \\ &= \frac{S_t}{M_t^{1-\rho}} M_t \\ &= S_t M_t^{\rho} \end{aligned} \tag{17}$$

where the second equality uses the software-capacity constraint, $S_t = s_{it} M_t^{1-\rho}$.

Thus, software services exceed the software capital stock by the factor M_t^{ρ}. Non-rivalry amplifies the effective amount of software that can be used in production, and this amplification is increasing in market size M_t, with strength governed by the degree of specificity ρ. If $\rho = 0$, then $X_t = S_t$ and there is no amplification. If $\rho \to 1$, then $X_t = S_t M_t$, meaning the software stock is fully shareable across firms, and the software services grow proportionally with the market size.

Aggregate output. The aggregate production function can be written in two equivalent ways that highlight the amplification from software non-rivalry:

$$\begin{aligned} Y_t &= M_t^{\psi} z_t \left[K_t^{\alpha_k} S_t^{\alpha_s} L_t^{\alpha_l} \right]^{\eta}, \qquad \psi \equiv 1 - \eta + \alpha_s \eta \rho, \\ Y_t &= M_t^{1-\eta} z_t \left[K_t^{\alpha_k} X_t^{\alpha_s} L_t^{\alpha_l} \right]^{\eta} \quad \text{since } X_t = M_t^{\rho} S_t. \end{aligned} \tag{18}$$

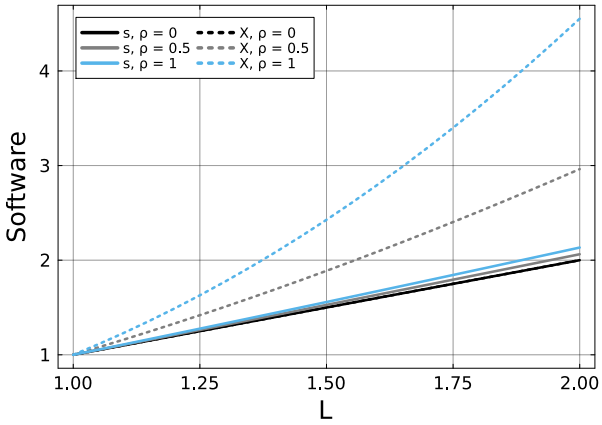
The first line shows that output increases with the mass of firms M_t. In a standard setup without non-rival software, the elasticity of output with respect to M_t is $1 - \eta$. With software, the elasticity becomes $\psi \equiv 1 - \eta + \alpha_s \eta \rho$. In other words, the elasticity depends on the degree of specificity but multiplied by the software share α_s. The non-rivalry only matters to the extent that software is an important factor of production.

The second line rewrites aggregate output in terms of software services X_t. The gap between the stock of software capital S_t and the flow of services X_t is what generates the additional scale effects. Writing aggregate output in terms of X_t yields a factor $M_t^{1-\eta}$, which is the usual scale effect in models with decreasing returns at the firm level.

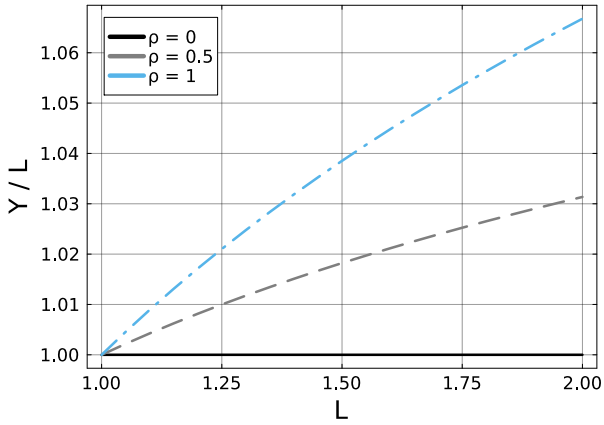
We next discuss two implications of non-rival software capital. First, software non-rivalry generates aggregate scale effects. Second, it amplifies the effects of software-specific technical change.

Figure 2: Scale Effects from Non-Rival Software

(A) Software stock and services



(B) Output per worker



3.2 Non-rival software capital generates scale effects

The first implication is that software non-rivalry generates aggregate scale effects of the type emphasized by Romer (1990) and Jones (1995). Because software can be reused, a larger economy can obtain more software services from a given stock of software capital.

To illustrate the scale effects, Figure 2 shows comparative statics with respect to the labor force, L. Panel (a) shows software capital, S, with solid lines and software services, X, with dashed lines. When $\rho = 0$, software capital and software services coincide, so the two black lines are indistinguishable. When $\rho > 0$, however, software services grow faster than the underlying software stock, and the gap between them is larger for higher values of ρ.[2]

Panel (b) shows the corresponding response of output per worker. When $\rho = 0$, software cannot be reused across firms, so there are no aggregate scale effects. Output per worker is independent of L, and the black line is flat. When $\rho > 0$, output per worker rises with the size of the labor force, and this response becomes stronger as ρ increases.

The elasticity of output per worker with respect to the labor force is

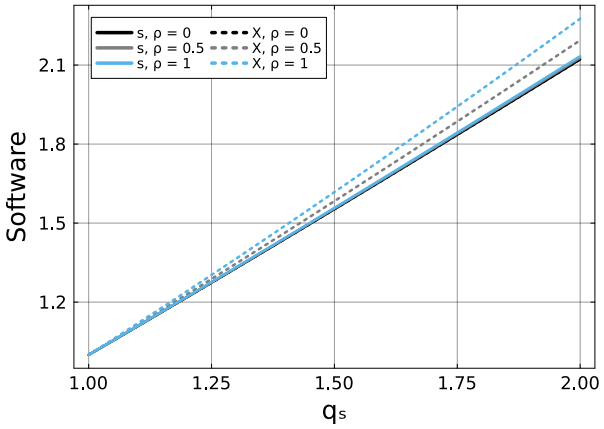
$$\frac{\partial \log(Y/L)}{\partial \log L} = \frac{\alpha_s \rho}{1 - \alpha_s - \alpha_k - \alpha_s \rho}.$$

The scale effect is increasing in both the software share, α_s, and the degree of shareability, ρ. In the denominator, $1 - \alpha_s - \alpha_k$ captures the amplification generated by traditional-capital and software deepening, while the term $-\alpha_s \rho$ captures the additional amplification generated by the expanding

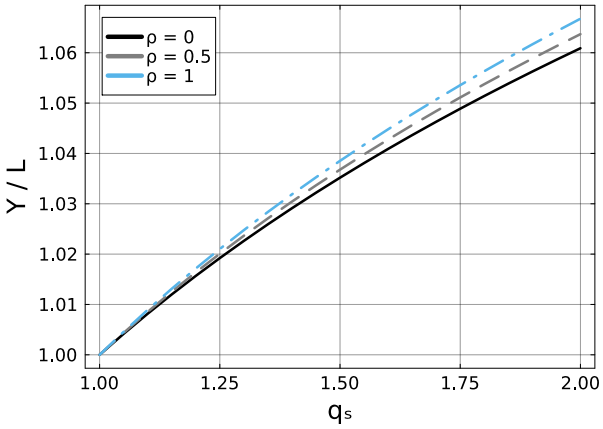
[2]This is true as long as $M > 1$. Software shareability increases services only when there are multiple users. This condition holds in all our calibrations. The elasticity of output per worker with respect to L, however, does not depend on the level of M.

Figure 3: The Impact of a Software Productivity Shock

(A) Software stock and services



(B) Output per worker



number of software users, M. Appendix A provides the full derivation.

3.3 Non-rival software capital amplifies software-specific technical change

Next, we look at the impact of a software-specific technical change shock, or a shock to q^s, when software is non-rival and non-specific.

Figure 3 presents comparative statics with respect to software-specific productivity, q^s. Panel (a) shows the responses of software capital, S, and software services, X. An increase in q^s allows software producers to create more software capital from a given amount of investment. The resulting increase in output further raises software investment and, in turn, the software stock. This feedback operates for all values of ρ, but is stronger when software is more shareable. When $\rho = 0$, software services coincide with the software stock, so the two black lines are indistinguishable. When $\rho > 0$, software services grow faster than the underlying stock because the additional software capital can be reused across users. Consequently, as Panel (b) shows, output per worker responds more strongly to q^s at higher values of ρ. Thus, software non-rivalry amplifies the effects of software-specific technical change.

The elasticity of output with respect to software productivity, q^s, is

$$\frac{\partial \log(Y/L)}{\partial \log q^s} = \frac{\alpha_s}{1-\alpha_k-\alpha_s-\alpha_s\rho}. \tag{19}$$

The effects of software-specific technical change are therefore stronger when software accounts for a larger share of production, as given by α_s, and increasing in shareability, ρ.

Where does this amplification come from? To identify the underlying channels, log-differentiate

the aggregate production function in Equation (18) with respect to q^s:

$$\frac{\partial \log Y}{\partial \log q^s} = \underbrace{\alpha_s \eta \frac{\partial \log S}{\partial \log q^s}}_{\text{direct effect and software deepening}} + \underbrace{\alpha_k \eta \frac{\partial \log K}{\partial \log q^s}}_{\text{capital deepening}} + \underbrace{(1-\eta+\alpha_s \eta \rho)}_{\text{scale elasticity}} \underbrace{\frac{\partial \log M}{\partial \log q^s}}_{\text{user-base response}}. \tag{20}$$

The first two terms capture software and traditional-capital deepening. Because investment in both types of capital is proportional to output, the initial increase in output raises both capital stocks, further increasing output. The final term captures the response of the user base, $\partial \log M / \partial \log q^s$, multiplied by the elasticity of aggregate output with respect to the number of users. An increase in M raises output through both the traditional span-of-control channel, $1-\eta$, and the reuse of non-rival software, $\alpha_s \eta \rho$. The latter channel is stronger when software represents a larger share of production and is more shareable.

Because K and M are proportional to Y, and S is proportional to $q^s Y$, substituting these relationships into Equation (20) yields the analytical expression in Equation (19). Appendix A provides the full derivation.

The key takeaway is that non-rivalry amplifies software-specific technical change only when higher software productivity increases the number of users, M.[3] This is intuitive because, by its nature, non-rivalry allows the same software capital to be shared across many users. As the user base expands, a given software stock can be used more broadly and therefore provides more software services. If the number of users does not respond, this additional amplification channel is absent. The same logic governs the response of output to L. A larger labor force generates scale effects from software non-rivalry to the extent that it expands the number of software users.

3.4 Efficiency

To assess the efficiency of the decentralized equilibrium, we compare its steady-state output and allocation of resources with the first-best allocation chosen by a social planner. The planner maximizes household utility in Equation (11) subject to the economy's production technologies and resource constraints. We provide the planner's problem and its optimality conditions in Appendix **??**.

The decentralized equilibrium features two sources of inefficiency. First, final-good firms do

[3] In the simple model, the response of M is sensitive to the denomination of the fixed cost. We relax this feature in the quantitative model by introducing an explicit adoption margin.

not internalize the benefits of their entry for software sharing, leading to insufficient entry relative to the planner's allocation:

$$\frac{M^D}{M^P} = \frac{1-\eta}{1-\eta+\eta\alpha_s\rho}\frac{Y^D}{Y^P} < 1. \tag{21}$$

The additional term $(\eta\alpha_s\rho)$ in the denominator captures the social benefit of entry from expanding the software user base. As more firms enter, a given software stock can serve more users, lowering the equilibrium software rental rate.

Second, monopolistically competitive software producers charge a markup over the marginal cost of providing software services, leading to underinvestment in software capital. In steady state, the ratio of software capital in the decentralized equilibrium to that in the planner's allocation is

$$\frac{S^D}{S^P} = \frac{\varepsilon-1}{\varepsilon}\frac{Y^D}{Y^P} < 1. \tag{22}$$

These two distortions result in lower aggregate output in the decentralized equilibrium:

$$\frac{Y^D}{Y^P} = \left[\left(\frac{1-\eta}{1-\eta+\eta\alpha_s\rho}\right)^{\frac{1-\eta+\eta\alpha_s\rho}{\eta}}\left(\frac{\varepsilon-1}{\varepsilon}\right)^{\alpha_s}\right]^{\frac{1}{1-\alpha_k-\alpha_s-\alpha_s\rho}} < 1. \tag{23}$$

The output gap widens as software becomes more shareable (higher ρ). Greater shareability strengthens the positive externality from firm entry and amplifies the aggregate consequences of software underinvestment due to monopolistic competition. Thus, while software non-rivalry amplifies the benefits of software-specific technical change, it also magnifies the output losses from inefficient firm entry and software investment.

4 Quantifying the Impact of Software-Specific Technical Change

In this section, we quantify the impact of software-specific technical change and the scale effects implied by non-rival software capital. Before outlining the specific exercise, we first enrich the simple model presented in Section 2 along two dimensions to make it appropriate for the quantitative exercise. We then describe the calibration of the model to match the data in 1997.

We use the model for growth accounting, similar to the exercise in Greenwood et al. (1997). Specifically, we ask how much growth in output per worker can be attributed to software-specific technical change q^s, investment-specific technical change q^k, labor force growth L, and a residual

Hicks-neutral productivity shock A.

4.1 Quantitative Model

The simple model presented in Section 2 made two strong assumptions that we relax in the quantitative model. First, we introduce an explicit adoption margin. Final-good firms choose between a technology that uses software and one that does not. As discussed in Section 3, the response of the number of software users is central to the amplification of software-specific technical change. In the simple model, the number of users, M, changes only through firm entry, and the strength of this response depends on how the fixed cost is denominated. The quantitative model instead allows the number of software users to respond directly through firms' adoption decisions.

Second, we allow software to be reused both across and within firms. In the simple model, the gains from non-rivalry arise only because the same software can be used by multiple firms. In practice, software can also be used simultaneously by multiple employees within a firm. The quantitative model incorporates both the across-firm and within-firm dimensions of software reuse.

The full economic environment for the quantitative model is given in Table 10. In this section, we describe the two main changes.

Adding an Adoption Margin. The first extension adds an explicit software-adoption decision. Final-good firms choose between two technologies: a traditional technology that uses capital and labor and a software technology that also uses software. Their production functions are

$$\begin{aligned} y_{it}^n &= A_t z_t^n \left(k_{it}^{\beta_k} l_{it}^{\beta_l}\right)^{\eta}, & \beta_k + \beta_l &= 1, \\ y_{it}^a &= A_t z_t^a \left(k_{it}^{\alpha_k} s_{it}^{\alpha_s} l_{it}^{\alpha_l}\right)^{\eta}, & \alpha_k + \alpha_s + \alpha_l &= 1, \end{aligned}$$

where s_{it} is the same software bundle defined in the simple model. The relative productivity of the adoption technology and non-adoption technologies are given by z^a and z^n, respectively. Because software is more expensive than traditional capital, for anyone to adopt, the adopter technology must be more productive than the non-adopter technology, $z_t^a > z_t^n$. We include a common Hicks-neutral productivity term, A_t, in both technologies. In the quantitative exercise, this term allows us to distinguish the contribution of Hicks-neutral productivity growth from those of the other forces driving aggregate growth.

Firms differ in their fixed costs of adoption, $f_i^a \sim G$, which are denominated in units of labor. A firm adopts when the resulting increase in gross profits exceeds its adoption cost:

$$\pi_t^a - \pi_t^n > w_t f_i^a,$$

where π_t^a and π_t^n denote gross profits under the adopting and traditional technologies. The mass of adopters is therefore

$$M_t^a = M_t G\left(\frac{\pi_t^a - \pi_t^n}{w_t}\right).$$

The free-entry condition must now account for both the endogenous technology choice and the heterogeneous adoption cost:

$$\mathbb{E}_i\left[\max\left\{\pi_t^n, \pi_t^a - w_t f_i^a\right\}\right] = w_t f_c.$$

We now denominate the fixed cost f_c in units of labor.[4]

The aggregate software share is

$$\lambda_t^s \equiv \frac{r_t^s X_t}{Y_t} = \eta\,\alpha_s \frac{Y_t^a}{Y_t}.$$

It depends on both the software share within adopting firms, $\eta\,\alpha_s$, and the share of aggregate output produced by adopters, Y_t^a/Y_t. In the simple model, all firms use software and the aggregate software share is fixed at $\eta\,\alpha_s$. Introducing adoption allows the importance of software in aggregate production to increase over time, consistent with the evidence in Figure 1.

Within-Firm Non-Rivalry. The second extension allows software to be reused within the firm. In the simple model, a given piece of software can be shared across firms but not across workers within a firm. We relax this assumption so that the same piece of software code can serve multiple workers. Specifically, the software producer must hold a stock sufficient to accommodate a nested CES-type aggregate of software use per worker. The resource constraint becomes:

$$s_{jt} \geq \left[\int_0^{M_t^a}\left(\int_0^{l_{it}^a}\left(\frac{s_{ijt}}{l_{it}^a}\right)^{\frac{1}{1-\rho_l}} d\ell\right)^{\frac{1-\rho_l}{1-\rho_m}} di\right]^{1-\rho_m},$$

where ρ_l governs the extent of reuse across workers within a firm and ρ_m governs reuse across firms.

As in the simple model, we assume that adopting final-good firms are identical. If we further

[4]This prevents the mass of firms from becoming unbounded as the economy grows, a problem that can arise when entry costs are denominated in units of the final good; see Klenow and Li (2025).

impose $\rho_l = \rho_m = \rho$, the capacity constraint simplifies to

$$s_{jt} \geq \underbrace{\frac{s_{ijt}}{l_{it}^a}}_{\text{software per worker}} \underbrace{(L_t^a)^{1-\rho}}_{\text{adopter labor}},$$

where $L_t^a = M_t^a l_{it}^a$ is total employment among adopting firms. Aggregating across software varieties, the relationship between the software stock produced by software firms, S_t, and the software services used in final-good production, X_t, becomes

$$X_t = S_t (L_t^a)^{\rho}.$$

The two models generate quantitatively similar amplification of software-specific technical change. In the simple model, amplification depends on the response of the mass of firms, $\partial \log M / \partial \log q^s$. In the quantitative model, it instead depends on the response of total adopter employment, $\partial \log L^a / \partial \log q^s$. Adopter employment can increase either because more firms adopt software or because existing adopters hire more workers. The latter intensive margin is quantitatively small in our exercise, reflecting the Cobb–Douglas production structure and our assumption that Hicks-neutral productivity, A_t, affects adopters and non-adopters proportionally. Although introducing within-firm reuse has only a modest quantitative effect, we add it because it removes the restrictive assumption that software cannot be shared across workers and to show how the same CES structure can capture both dimensions of non-rivalry.

4.2 Calibration

We next describe the model's calibration before turning to the growth-accounting exercise in Section 5. We calibrate the model's initial steady state to match the 1997 data. Our objective is to quantify the effects of software-specific technical change, represented by changes in q^s, and to examine how those effects vary with the degree of software specificity, ρ. Because ρ directly affects the software rental rate, and therefore the model's steady state, changing ρ while holding the remaining parameters fixed would also change the initial calibration. We therefore recalibrate the model to the same 1997 targets for each value of ρ and then apply the same sequence of shocks to each calibrated economy.

Table 1 presents the full calibration for $\rho = 0$. The corresponding calibration tables for $\rho = 0.5$ and $\rho = 1$ are reported in the appendix.

We next describe the calibration of the individual parameters. We set δ_k equal to the average

Table 1: Calibration with $\rho = 0$

Param.	Description	Value	Source	Data	Model
β	Discount factor	0.960	–	–	–
δ_k	Depreciation k	0.132	BEA	0.132	0.132
δ_s	Depreciation sw	0.330	BEA	0.330	0.330
η	DRTS	0.850	Profit share, Barkai	0.150	0.150
β_l	Non-adopter labor exponent	0.561	BEA labor and software shares	0.605	0.605
α_l	Adopter labor exponent	0.510	$\beta_l - \alpha_s$	–	–
β_k, α_k	Capital exponent	0.439	$1 - \beta_l$	–	–
α_s	Software share adopters	0.051	BEA agg. sw share	0.022	0.022
z_a	Productivity adopters	1.175	Rel. adopter emp., ACES	1.400	1.400
f_c	Fixed cost	0.003	Production/software firm count, BDS	81.132	81.132
q_k	ISTC	1.467	K/Y, BEA–BLS	3.160	3.160
q_s	SW-STC	0.543	X/Y, BEA–BLS	0.024	0.024
μ_a	Mean of log adoption cost	-6.324	Adoption share, ACES	0.400	0.400
σ_a	Sd of log adoption cost	0.896	SW-share–r^s log slope	-0.731	-0.731

Notes: z_n, L, and the mass of software-producing firms are normalized to one.

depreciation rate for equipment, weighting the depreciation rate of each asset by its current-cost capital stock. The asset-specific depreciation rates are taken from Caunedo et al. (2023), who in turn gets them from the BEA. We set the software depreciation rate to $\delta_s = 0.33$, the rate used by the BEA for own-account software. We view own-account software as the closest empirical counterpart to the software capital accumulated by software producers through the law of motion for s_j.

We set the final-good firms' decreasing-returns parameter to $\eta = 0.85$. Under free entry, this implies a profit share of $1 - \eta = 0.15$, consistent with the profit share estimated by Barkai (2020).

We calibrate the software, traditional-capital, and labor shares jointly. The software share parameter, α_s, is chosen to match the aggregate software share. Because the aggregate software share also depends on the fraction of firms that adopt software and the relative size of adopters, α_s must be determined jointly with the parameters governing these two margins.

We assume that adopters and non-adopters have the same traditional-capital share $\alpha_k = \beta_k$. The labor shares are then the residuals, $\beta_l = 1 - \beta_k$, and $\alpha_l = 1 - \alpha_k - \alpha_s$. Conditional on α_s, this leaves one free parameter, β_k, which we choose to match the aggregate labor share. This restriction implies that adopters' software share comes at the expense of their labor share rather than their traditional-capital share. Adopters therefore have a lower labor share than non-adopters, while both groups have the same traditional-capital share. As the fraction of adopting firms increases, the aggregate software share rises and the aggregate labor share falls.

In the national accounts, some software expenditures are capitalized, while others are treated as intermediate expenses. In the model, however, the software share should include all payments for software services, regardless of their accounting treatment. We therefore correct the aggregate

software share from the BEA–BLS Industry-Level Production Account to reflect these additional expensed software payments. Specifically, we use the BEA input–output tables to estimate intermediate purchases from software-producing industries.[5] Let E_t^{sw} denote these expensed software purchases. The adjusted aggregate software share is then

$$\text{Adjusted SW Share}_t = \frac{\mu_t r_t^s X_t + E_t^{sw}}{VA_t + E_t^{sw}},$$

where $r_t^s X_t$ is measured compensation to capitalized software. μ_t adjusts the measured capital payments for the profit share.[6] Labor compensation is unchanged in levels, but its share is computed using the software-inclusive denominator: $\frac{w_t L_t}{VA_t + E_t^{sw}}$. These adjustments increase the measured software share from 0.015 to 0.022 in 1997 and from 0.033 to 0.045 in 2024. This is a strong assumption, since we treat all intermediate purchases assigned to the detailed software-producing industries as payments for software services, but it allows us to capture expensed software use that is omitted from measured software capital compensation.

The calibrated value of α_s is 0.051. This is a crucial parameter for the impact of software-specific technical change. Importantly, this parameter is the same across all calibrations of ρ. Thus, the amplification from non-rivalry is not driven by differences in α_s.

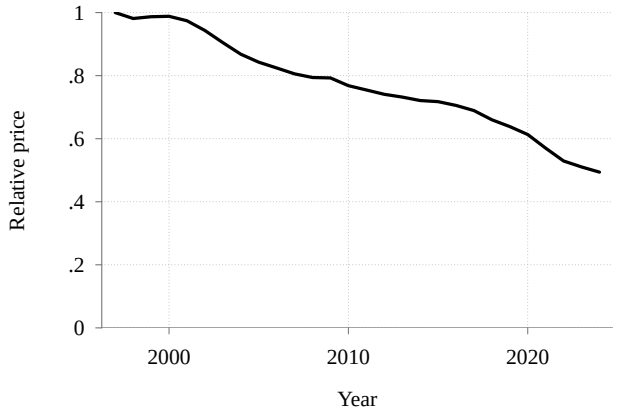
We assume that the fixed costs of adoption follow a lognormal distribution, $\log f_i^a \sim \mathcal{N}(\mu_a, \sigma_a)$. We normalize the productivity of non-adopters to $z^n = 1$ and jointly choose the relative productivity of adopters, z^a, and the mean log adoption cost, μ_a, to match the relative size of adopting firms and the fraction of firms that adopt. We normalize the mass of software producers to one and choose the fixed cost of final-good firms, f_c, to match the ratio of non-software firms to software-producing firms in the Business Dynamics Statistics. We define software producers as firms in NAICS 5112 (Software Publishers) and NAICS 5415 (Computer Systems Design and Related Services); all other firms are classified as final-good producers.

A key parameter is σ_a, the standard deviation of log adoption costs. This parameter governs how strongly the fraction of adopting firms responds to a decline in the price of software. As discussed in Section 3, the change in the number of software users is a key margin for the ampli-

[5] We define software-related intermediate purchases using the detailed BEA Use Table industries corresponding to software publishers (code 511200); data processing, hosting, and related services (518200); custom computer programming services (541511); computer systems design services (541512); and other computer-related services, including facilities management (54151A). We map these detailed industries to the annual BEA summary industry codes 511, 514, and 5415.

[6] The BEA-BLS TFP framework assumes that labor and capital compensation add up to value added. Capital compensation is generally calculated as a residual after subtracting labor compensation. We therefore treat pure profits as part of measured capital compensation. To remove the target profit share, we scale software and traditional-capital payments proportionally by $\mu_t \equiv \frac{r_t^s X_t + r_t^k K_t - (1-\eta) VA_t^{sw}}{r_t^s X_t + r_t^k K_t}$.

Figure 4: Relative Price of Own-Account Software



fication of software-specific technical change from non-rivalry. We do not have direct evidence that identifies the structural distribution of adoption costs. Moreover, even estimates for particular technologies would not necessarily map cleanly into our framework because the adoption margin in our stylized model is intended to capture broader changes in production that raise the software share and reduce the labor share over time.

We therefore calibrate σ_a to match the historical relationship between the aggregate software share and the software rental rate. Specifically, we estimate a time-series regression of the log software share on the log software rental rate. The rental rate is measured using a Törnqvist index that combines the rental rates of pre-packaged, vendor-customized, and own-account software. For the model counterpart, we perturb software productivity, q^s, and numerically calculate the change in the log software share relative to the change in the log software rental rate. We choose σ_a so that this model-implied elasticity matches its empirical counterpart.

Finally, we choose q^s and q^k to match the empirical ratios of effective software capital and traditional capital to output, X/Y and K/Y, respectively. We measure traditional capital using the BEA-BLS non-software productive-capital stock. As with the software share, we adjust the software stock from the BEA-BLS to account for intermediate payments. Effective software capital includes both capitalized software and a stock-equivalent measure of expensed software, $X_t = X_t^{\text{cap}} + E_t^{sw}/r_t^s$, where r_t^s is the rental rate for pre-packaged software. Both ratios use software-inclusive value added, $VA_t + E_t^{sw}$, expressed in 2017 consumption dollars in the denominator.

5 Growth Accounting

In this section, we use the model to attribute aggregate growth between 1997 and 2024 into contributions from software-specific technical change, q^s, investment-specific technical change, q^k, labor

Table 2: Decline in p^s vs. r^s

	Software		
ρ	$\%\Delta q_s$	$\%\Delta p_s$	$\%\Delta r_s$
0.0	102.5%	-50.6%	-50.6%
0.5	102.5%	-50.6%	-59.6%
1.0	102.5%	-50.6%	-68.8%
Data	–	-50.6%	-75.8%

force growth, L, and residual Hicks-neutral productivity growth, A. We start by examining the impact of software-specific technical change in isolation before looking at the full growth-accounting exercise in Section 5.2.

5.1 Software-Specific Technical Change

We start by looking at the impact of a software-specific technical change shock in isolation. Software-specific technical change refers to improvements in the efficiency with which the final good can be converted into software capital, given by q^s.

We identify changes in q_t^s using the decline in the relative price of own-account software investment shown in Figure 4. This follows the approach of Greenwood et al. (1997), who use the relative price of new equipment to identify investment-specific technical change. We use the own-account software price because it most closely corresponds to the cost of producing the new software capital accumulated by software producers. In contrast, the rental rate r_t^s is the endogenous price paid by final-good firms for software services. The own-account price index is constructed partly from input costs, including the compensation of software developers, and therefore more closely maps to changes in the cost of producing new software capital.[7]

The relative price of own-account software fell by 51% between 1997 and 2024, implying that the productivity of producing new software capital approximately doubled. In the quantitative model, the steady-state software rental rate is

$$r_t^s = \frac{\varepsilon}{\varepsilon - 1} \frac{\frac{1}{q_t^s}\left(\frac{1}{\beta} - 1 + \delta_s\right)}{(L_t^a)^\rho}.$$

The response of the rental rate therefore depends on the degree of specificity, ρ. Table 2 reports the exogenous change in the software investment price and the resulting endogenous change in the

[7] It actually averages the cost of inputs and the pre-packaged software series, making it imperfect for our exercise, but the best available.

Table 3: Growth in S vs. X

ρ	$\Delta\log(X/L)$	$\Delta\log(S/L)$	$\rho\,\Delta\log L^a$
0.0	108.5%	108.5%	0.0%
0.5	135.4%	115.2%	20.2%
1.0	168.1%	122.2%	45.9%

$\Delta\log(X/L) = \Delta\log(S/L) + \rho\,\Delta\log L^a$.

Table 4: Growth Accounting: Software Specific Technical Change

		Contribution to growth in Y/L (pps)			Contribution to X/L (pps)	
ρ	$\%\Delta(Y/L)$	TFP	K/L	X/L	Stock S/L	Reuse
0.0	4.75%	0.07	1.75	2.93	2.93	0.00
0.5	6.19%	0.14	2.26	3.79	3.22	0.57
1.0	8.07%	0.27	2.92	4.88	3.55	1.33

rental rate for each value of ρ.

When $\rho = 0$, software is fully specific and the rental rate moves one-for-one with the investment price, $p_t^s = 1/q_t^s$. Both fall by 51%. When $\rho > 0$, the increase in adopter employment, L_t^a, further reduces the rental rate, causing it to fall by more than the investment price. Although this moment is not targeted, the model with $\rho = 1$ generates a 69% decline in the rental rate, close to the 76% decline in the rental rate of pre-packaged software observed in the data. Software non-rivalry therefore provides a mechanism through which the market rental rate of pre-packaged software can fall more sharply than the cost of producing own-account software, even though the underlying technology for producing them is the same.

Table 3 shows the impact of the software-specific technical change shock on software services X versus the software capital stock S. With no reuse, $\rho = 0$, the two are the same. As software productivity approximately doubles, both S/L and X/L increase by approximately 109 log points.

When $\rho > 0$, the software stock grows somewhat more. The larger decline in the rental rate encourages adoption, raises the aggregate software share, and increases demand for software services, inducing software producers to accumulate additional software capital. Consequently, S/L increases by 122 log points when $\rho = 1$, compared with 109 log points when $\rho = 0$. The larger difference, however, comes from reuse. Because $X_t = S_t(L_t^a)^\rho$, growth in adopter employment allows a given software stock to provide more services. When $\rho = 1$, X/L increases by 168 log points, of which 46 log points are attributable to the growth in the reuse term.

Finally, Table 4 presents the decomposition of growth in output per worker into contributions from TFP, traditional-capital deepening, and software deepening. The contributions of K/L and X/L are the standard capital-deepening terms, adjusted to account for differences between adopters

Table 5: Calibration of full shock

Param.	Description	Value	Target	Data
q_T^s/q_0^s	SW-STC	2.025	Decline in rel. price of O.A. SW	0.494
q_T^k/q_0^k	ISTC	1.063	Decline in rel. price of capital	0.941
L_T/L_0	Growth in labor force	1.233	Labor force growth	1.233
A_T/A_0	Residual productivity	1.185–1.221	Growth in Y/L	1.531

and non-adopters. TFP is measured as the residual after accounting for traditional-capital and software deepening. It therefore captures both the direct productivity gains associated with adoption, the effects of reallocating capital, software, and labor between adopters and non-adopters, and the traditional span-of-control effect from the decreasing-returns-to-scale production function. Appendix B.1 provides the precise definition of each component.

The software-specific technical-change shock increases output per worker by 8.07% when $\rho = 1$, compared with 4.75% in the rival benchmark, $\rho = 0$. The rival benchmark therefore understates the effect of software-specific technical change by 3.32 percentage points, or approximately 70%. When $\rho = 1$, software services account for 4.88 percentage points of output growth, with 3.55 percentage points arising from growth in the software stock and 1.33 percentage points from reuse. Thus, even if the software stock were measured correctly, ignoring reuse would miss approximately 27% of the total contribution from software services.

The software-specific technical-change shock also generates traditional-capital deepening. As output grows, a constant fraction of output is devoted to traditional capital which also rises. Because growth is higher when $\rho = 1$, the contribution from capital deepening is also higher, increasing from 1.75pps in the rival benchmark to 2.92pps when $\rho = 1$. The shock also induces growth attributed to the TFP term, though the contribution is small. Because adopters are more productive than non-adopters, increased adoption raises average productivity and reallocates capital, software, and labor across the two technologies.

5.2 Full Growth Accounting Exercise

The previous section considered software-specific technical change in isolation. We now turn to the full growth-accounting exercise, which decomposes growth between 1997 and 2024 into contributions from software-specific technical change, q^s, investment-specific technical change, q^k, labor-force growth, L, and residual Hicks-neutral productivity growth, A.

Table 5 shows the calibration of the full shock. Software productivity increases by 102.5%, the same as in Section 5.1. We increase q^k by 6.3% to match the 5.9% decline in the relative price of

Table 6: Growth Accounting—Share of growth attributed to each shock

		Share of Y/L gr. attributed to:			
ρ	$\%\Delta(Y/L)$	A	L	q_k	q_s
0.0	53.1%	79.9%	0.0%	9.1%	11.0%
0.5	53.1%	75.3%	2.1%	9.3%	13.4%
1.0	53.1%	69.9%	5.1%	9.6%	15.4%
Missed by rival benchmark (pps)	–	-9.96	5.11	0.44	4.40

non-software capital.[8] We also increase L by 23.3%, matching growth in the civilian labor force reported by the BLS.

For each value of ρ, we choose the residual Hicks-neutral productivity shock to reproduce the 53.1% increase in real GDP per civilian labor-force participant over this period. The observed changes in q^s, q^k, and L generate more aggregate growth when software is more shareable. Consequently, the residual increase in A required to match observed growth declines with ρ: it falls from 22.1% in the rival benchmark to 18.5% when $\rho = 1$.

Table 6 reports the share of the 53.1% increase in output per worker attributable to each shock. Because the shocks interact, their measured contributions may depend on the order in which they are introduced. We therefore use the Shapley–Owen–Shorrocks decomposition, which provides an ordering-invariant attribution (Audoly et al., 2025; Shorrocks, 2013).[9]

The Hicks-neutral productivity shock is the largest contributor to growth, accounting for 79.9% in the rival benchmark. Its contribution declines as ρ rises, falling by 10.0 percentage points to 69.9% when software is fully shareable.

By contrast, the contribution of software-specific technical change rises from 11.0% when $\rho = 0$ to 15.4% when $\rho = 1$. The contribution of labor-force growth also increases with ρ. When $\rho = 0$, there are no aggregate scale effects, so labor-force growth does not raise output per worker and contributes nothing to its growth. When $\rho = 1$, a larger workforce expands the user base over which software can be reused, allowing software services to grow faster than the underlying software stock. Labor-force growth therefore accounts for 5.1% of growth, reducing the share attributed to residual Hicks-neutral productivity.

Together, software-specific technical change and the scale effects generated by labor-force growth account for 20.5% of growth when software is fully shareable, compared with 11.0% in

[8] The relatively small change in the aggregate capital price reflects the fact that we combine equipment and structures. Between 1997 and 2024, the relative price of equipment fell by 50.9%, whereas the relative price of structures increased by 62.6%.

[9] For each shock, the decomposition averages its marginal contribution over all possible orderings of the shocks.

the rival benchmark. Thus, the rival benchmark misses just under half of the growth attributable to these two channels.

Table 7 decomposes growth into capital deepening, software deepening, and a residual TFP term using the growth-accounting framework described in Section 5.1 and Appendix B.1. This differs from Table 6, which attributes growth to the four fundamental shocks. Table 7 provides a decomposition by factor, whereas Table 6 provides a decomposition by underlying shock. For example, in the exercise with only software-specific technical change, all the growth is ultimately attributed to that shock, even though its effects operate through TFP, traditional capital deepening, and software deepening.

The contribution from software services increases from 5.4pps in the rival benchmark with $\rho = 0$ to 8.89pps in the fully shareable case, $\rho = 1$. Thus, the rival benchmark underestimates the contribution of software by 3.48pps or 64%. The contribution of the TFP term falls from 25.6pps to 22.7pps.

Table 7: Growth Accounting—Contribution of each factor to growth

		Contribution to growth in Y/L (pps)			Contribution to X/L (pps)	
ρ	$\%\Delta(Y/L)$	TFP	K/L	X/L	Stock S/L	Reuse
0.0	53.10%	25.64	22.05	5.41	5.41	0.00
0.5	53.10%	24.25	21.79	7.06	5.78	1.28
1.0	53.10%	22.70	21.51	8.89	6.10	2.79

6 Software in the National Accounts

Software non-rivalry raises a measurement question: what is the relevant object to record as software capital? In our model, there is a distinction between the stock of software capital produced by software firms, S_t, and the flow of software services used in production, X_t. These objects coincide when software is rival, but differ when software can be reused. In the quantitative model, $X_t = S_t(L_t^a)^\rho$, so that a given software stock can generate more services as it is deployed across a larger set of users.

This distinction matters for growth accounting. Measuring the productive software input using X_t attributes the gains from reuse to software deepening. Measuring instead the underlying stock S_t attributes those same gains to TFP. Neither treatment changes measured output; it only changes the decomposition of output growth between factor deepening and productivity.

However, the BEA's treatment of software introduces an additional measurement issue. Our model assumes that software producers incur investment expenditures I_t^s to accumulate the under-

lying software stock and then charge final-good firms for access to the services generated by that stock. This formulation is particularly natural for software-as-a-service (SaaS). For example, we interpret a subscription payment to a software firm (e.g. SAP) as a payment for services generated by software capital accumulated by SAP. Conceptually, such a subscription differs from the purchase of a long-lived software license.[10] In practice, however, this distinction is difficult to implement in the national accounts. The BEA constructs prepackaged software investment largely from the revenues of software publishers and related providers, and SaaS revenues are included in this measure. The underlying revenue data therefore do not, in general, identify whether a particular payment reflects a short-term subscription or the purchase of a long-lived license.[11] At the same time, expenditures by software producers to develop software originals are capitalized as R&D. Thus, in the SAP example, the national accounts can record investment both when SAP develops the underlying software original (R&D investment) and when customers acquire software from SAP (prepackaged software investment), whereas in our model the latter is a current payment for services generated by the former.

We use the model to quantify how these alternative measurement choices affect growth accounting. Specifically, we revisit the software-specific technical change exercise from Section 5.1, holding the underlying model allocation fixed and varying only the measurement of software. We consider three treatments. First, we measure software by the services used in production, X_t. Second, we measure only the underlying software stock, S_t. In both cases, measured GDP is the model's output, Y_t. The difference between the two treatments is therefore purely one of attribution: growth in $X_t/S_t = (L_t^a)^\rho$ is recorded as capital deepening when X_t is measured and as TFP growth when S_t is measured.

Our third treatment approximates the BEA's national-accounts treatment by capitalizing both producer-side software development and measured software purchases by final-good firms. Producer-side software investment is already incorporated in the model through the accumulation of S_t. We additionally treat payments by final-good firms for software, $r_t^s X_t$, as investment. Under this treatment, measured GDP is

$$Y_t^{\text{BEA}} = Y_t + r_t^s X_t. \tag{24}$$

Thus, unlike the distinction between measuring X_t and S_t, this accounting treatment changes mea-

[10] Even under a long-lived license, the software producer retains the underlying software original and can continue licensing it to other users, making it different from traditional rival capital.

[11] BEA's software methodology distinguishes conceptually between short-lived licenses and long-lived software acquisitions, but notes that annual licenses are not separately identified in purchased-software receipts. More recent descriptions of the methodology explicitly include software-as-a-service revenues in prepackaged software investment.

Table 8: Software Measurement and Growth Accounting

Treatment	$\rho = 0$			$\rho = 1$		
	Output (%)	TFP (pp)	Capital (pp)	Output (%)	TFP (pp)	Capital (pp)
X measured	4.75	0.07	4.68	8.07	0.27	7.80
S measured	4.75	0.07	4.68	8.07	1.60	6.47
BEA: Capitalize S and X	5.65	-1.24	6.89	9.36	-1.23	10.59

Notes: The table compares growth-accounting decompositions under alternative treatments of software. X denotes software services and S the underlying software stock. The BEA-style treatment capitalizes producer-side software development and measured software purchases by final-good firms. The allocation is held fixed across treatments.

sured output itself.[12]

Capitalizing software purchases also changes measured factor shares and the amount of capital deepening attributed to software. We compute the associated TFP residual as

$$\Delta \log \mathrm{TFP}_t^{\mathrm{BEA}} = \Delta \log \left(\frac{Y_t^{\mathrm{BEA}}}{L_t} \right) - \bar{v}^{k,\mathrm{BEA}} \Delta \log \left(\frac{K_t}{L_t} \right) - \bar{v}^{x,\mathrm{BEA}} \Delta \log \left(\frac{X_t}{L_t} \right) - \bar{v}^{s,\mathrm{BEA}} \Delta \log \left(\frac{S_t}{L_t} \right), \tag{25}$$

where the $\bar{v}$'s denote the corresponding average measured cost shares.

Table 8 reports the results. When software is fully specific ($\rho = 0$), $X_t = S_t$, so measuring software services or the underlying stock yields the same decomposition. The software-specific productivity shock raises output per worker by 4.75 percent, with 4.68 percentage points attributed to capital deepening and only 0.07 percentage points to TFP.

The distinction becomes important when software is shareable. With $\rho = 1$, the same shock raises output per worker by 8.07 percent. When software services X_t are measured, 7.80 percentage points are attributed to capital deepening and 0.27 percentage points to TFP. Measuring instead the underlying software stock S_t leaves output growth unchanged but reduces the contribution of capital deepening to 6.47 percentage points and raises measured TFP growth to 1.60 percentage points. The difference is 1.33 percentage points—exactly the contribution of software reuse identified in our baseline decomposition. Hence, whether the gains from non-rival software appear as capital deepening or as TFP growth depends on whether the measured input captures the flow of software services or the underlying software stock.

[12]The BEA takes the investments in pre-packaged software and accumulates them via perpetual inventory method to get the current pre-packaged software stock. To keep things simple, here we assume that the depreciation rate for pre-packaged software is 1 so that the investment flows $r_t^s X_t$ give the stock X_t after adjusting for the price.

The national-accounts treatment has a further effect because it changes measured output. For $\rho = 1$, capitalizing software payments raises measured output growth from 8.07 to 9.36 percent. At the same time, measured capital deepening rises to 10.59 percentage points, exceeding measured output growth and leaving a TFP residual of -1.23 percentage points. A similar pattern arises when $\rho = 0$. The negative residual reflects the fact that the BEA-style treatment capitalizes more software expenditure as investment, raising the measured software capital share and therefore the amount of growth attributed to capital deepening. Because the underlying allocation is unchanged, this decline in measured TFP is purely an accounting effect rather than an economic deterioration in productivity.

The broader implication is that the accounting treatment of non-rival software affects more than the measured stock of intangible capital. It determines whether the gains from reuse appear as capital deepening or TFP growth and, when software expenditures are capitalized, can also affect the measured growth rate of GDP itself and the resulting TFP growth.

7 Concluding Remarks

This paper develops a framework to study software-specific technical change in an economy where software is non-rival but subject to varying degrees of specificity. The key mechanism is that non-rivalry creates a gap between the stock of software capital and the flow of software services used in production. The magnitude of this gap depends on market size and the extent to which firms' software needs overlap.

Relative to a standard model that treats software as rival capital, our framework implies that software-sector productivity growth can generate amplified aggregate effects, particularly when software is less specific, market size is large, and software is an important component of aggregate output. In the full shareability case, our main growth-accounting exercise implies that software-specific technical change and the scale effects of non-rival software capital account for 21% of output-per-worker growth between 1997 and 2024. Modeling software as rival would miss nearly half of the contribution from these channels.

References

AUDOLY, R., R. MCGEE, S. OCAMPO, AND G. PAZ-PARDO (2025): "A Practitioner's Note on the Shapley–Owen–Shorrocks Decomposition," Staff Reports 1163, Federal Reserve Bank of New York.

BARKAI, S. (2020): "Declining Labor and Capital Shares," *The Journal of Finance*, 75, 2421–2463.

BRYNJOLFSSON, E., D. ROCK, AND C. SYVERSON (2021): "The Productivity J-Curve: How Intangibles Complement General Purpose Technologies," *American Economic Journal: Macroeconomics*, 13, 333–372.

CAUNEDO, J., D. JAUME, AND E. KELLER (2023): "Occupational Exposure to Capital-Embodied Technical Change," *American Economic Review*, 113, 1642–85.

CORRADO, C., C. HULTEN, AND D. SICHEL (2009): "Intangible Capital and U.S. Economic Growth," *Review of Income and Wealth*, 55, 661–685.

CROUZET, N., J. C. EBERLY, A. L. EISFELDT, AND D. PAPANIKOLAOU (2023): "Intangible Capital, Non-Rivalry, and Growth," NBER Working Paper 30987, National Bureau of Economic Research.

FISHER, J. D. M. (2006): "The Dynamic Effects of Neutral and Investment-Specific Technology Shocks," *Journal of Political Economy*, 114, 413–451.

GREENWOOD, J., Z. HERCOWITZ, AND P. KRUSELL (1997): "Long-run implications of investment-specific technological change," *The American economic review*, 342–362.

GREENWOOD, J. AND B. JOVANOVIC (1999): "The IT Revolution and the Stock Market," *American Economic Review*, 89, 116–122.

JONES, C. I. (1995): "R&D-Based Models of Economic Growth," *Journal of Political Economy*, 103, 759–784.

JOVANOVIC, B. AND P. L. ROUSSEAU (2005): "General Purpose Technologies," in *Handbook of Economic Growth*, ed. by P. Aghion and S. N. Durlauf, Elsevier, vol. 1B, 1181–1224.

KLENOW, P. J. AND H. LI (2025): "Entry Costs Rise with Growth," *Journal of Political Economy Macroeconomics*, 3, 43–74.

MCGRATTAN, E. R. (2020): "Intangible Capital and Business Cycles," *Journal of Political Economy*, 128, 1615–1655.

ROMER, P. M. (1990): "Endogenous Technological Change," *Journal of Political Economy*, 98, S71–S102.

SHORROCKS, A. F. (2013): "Decomposition Procedures for Distributional Analysis: A Unified Framework Based on the Shapley Value," *The Journal of Economic Inequality*, 11, 99–126.

Online Appendix

A Model Appendix

This appendix derives the results for the simple model in Section 2, whose economic environment is summarized in Table 9. The quantitative model in Section 4, which adds an adoption margin and within-firm reuse of software, is solved numerically.

A.1 Aggregate Production Function

The firm-level production function is

$$y_i = z_i \left(k_i^{\alpha_k} s_i^{\alpha_s} l_i^{\alpha_l}\right)^{\eta}, \qquad \alpha_k + \alpha_s + \alpha_l = 1, \quad \eta < 1,$$

where k_i is traditional capital, l_i is labor, and $s_i = \left(\int_0^1 s_{ij}^{\frac{\varepsilon-1}{\varepsilon}} dj\right)^{\frac{\varepsilon}{\varepsilon-1}}$ is the bundle of software services.

Traditional capital and labor are rival, so with a mass M of symmetric final-good firms, each firm uses $k_i = \frac{K}{M}$ and $l_i = \frac{L}{M}$, where K and L are the aggregate stocks of capital and labor. Software is non-rival but subject to specificity. By the resource constraint for software, each firm gets $s_{ij} = \frac{s_j}{M^{1-\rho}}$ units of service from producer j's stock s_j. Aggregating over varieties,

$$s_i = \left(\int_0^1 s_{ij}^{\frac{\varepsilon-1}{\varepsilon}} dj\right)^{\frac{\varepsilon}{\varepsilon-1}} = \frac{1}{M^{1-\rho}} \left(\int_0^1 s_j^{\frac{\varepsilon-1}{\varepsilon}} dj\right)^{\frac{\varepsilon}{\varepsilon-1}} \equiv \frac{S}{M^{1-\rho}},$$

where S is the CES bundle of software capital stocks. Because software producers are symmetric and have unit mass, s_j is the same for all j, so $S = s_j = \int_0^1 s_j\, dj$ also equals the aggregate software stock in the main text.

By symmetry, $z_i = z$ for all i, and output per firm is

$$\bar{y} \equiv \frac{Y}{M} = z\left[\left(\frac{K}{M}\right)^{\alpha_k} \left(\frac{S}{M^{1-\rho}}\right)^{\alpha_s} \left(\frac{L}{M}\right)^{\alpha_l}\right]^{\eta} = z M^{-\eta[\alpha_k + \alpha_s(1-\rho) + \alpha_l]} K^{\eta\alpha_k} S^{\eta\alpha_s} L^{\eta\alpha_l}.$$

Since $\alpha_k + \alpha_s + \alpha_l = 1$, the exponent on M simplifies to $-\eta(1-\alpha_s\rho)$, and multiplying by M gives

the aggregate production function

$$Y = zM^{\psi}\left(K^{\alpha_k}S^{\alpha_s}L^{\alpha_l}\right)^{\eta}, \qquad \text{where } \psi \equiv 1-\eta+\eta\alpha_s\rho. \tag{A.1}$$

Having more firms in the economy raises output through two channels, captured by the exponent ψ. The $1-\eta$ part is the benefit of spreading inputs across more firms when each firm has decreasing returns to scale. The $\eta\alpha_s\rho$ part comes from the non-rivalry of software: the same software stock delivers more total service when more firms use it.

A.2 Steady-State Equilibrium

We now solve for the steady state. The blocks below express the endogenous variables as functions of aggregate output Y and parameters. Substituting these expressions into the aggregate production function (A.1) then pins down Y.

Final-good firms. Taking the rental rate of capital r^k, the rental rates of software varieties $\{r_j^s\}$, and the wage w as given, firm i solves

$$\max_{k_i,\{s_{ij}\},l_i} z_i\left(k_i^{\alpha_k}s_i^{\alpha_s}l_i^{\alpha_l}\right)^{\eta} - r^k k_i - \int_0^1 r_j^s s_{ij}\,dj - wl_i, \qquad s_i = \left(\int_0^1 s_{ij}^{\frac{\varepsilon-1}{\varepsilon}}\,dj\right)^{\frac{\varepsilon}{\varepsilon-1}}.$$

Cost minimization across varieties gives the demand for variety j, $s_{ij} = s_i\left(r_j^s/R^s\right)^{-\varepsilon}$, where $R^s = \left(\int_0^1 (r_j^s)^{1-\varepsilon}dj\right)^{\frac{1}{1-\varepsilon}}$ is the rental-rate index for software, and total software spending is $\int_0^1 r_j^s s_{ij}\,dj = R^s s_i$. The first-order conditions for k_i, s_i, and l_i are given by

$$r^k k_i = \eta\alpha_k y_i, \qquad R^s s_i = \eta\alpha_s y_i, \qquad w l_i = \eta\alpha_l y_i. \tag{A.2}$$

Because the production function is Cobb–Douglas, spending on each input is a constant share of the firm's output. Combining the software condition with the demand for each variety gives

$$s_{ij} = \tilde{A}_i\left(r_j^s\right)^{-\varepsilon}, \qquad \tilde{A}_i \equiv \eta\alpha_s y_i\left(R^s\right)^{\varepsilon-1}. \tag{A.3}$$

Because total input costs equal ηy_i, the firm's operating profit is

$$\pi_i = y_i - r^k k_i - R^s s_i - wl_i = (1-\eta)\,y_i. \tag{A.4}$$

Varieties are symmetric, so $r_j^s = R^s \equiv r^s$. Multiplying (A.2) by the mass of firms M and using $Y = My_i$, $K = Mk_i$, $L = Ml_i$, and total software services $X \equiv Ms_i$,

$$K = \frac{\eta\,\alpha_k}{r^k}\,Y, \qquad X = \frac{\eta\,\alpha_s}{r^s}\,Y, \qquad L = \frac{\eta\,\alpha_l}{w}\,Y. \tag{A.5}$$

Software services and the software stock are linked through the resource constraint for software: $s_i = S/M^{1-\rho}$, so

$$X = Ms_i = M^{\rho} S. \tag{A.6}$$

Software producers. Producer j rents its stock s_{jt} to all M_t firms. Combining the resource constraint for software, $s_{jt} = s_{ijt} M_t^{1-\rho}$, with the demand curve (A.3) gives the inverse demand, where $\tilde{A}_t$ is the demand shifter $\tilde{A}_{it}$ common to all symmetric firms:

$$r_{jt}^s = \tilde{A}_t^{1/\varepsilon}\, M_t^{(1-\rho)/\varepsilon}\, s_{jt}^{-1/\varepsilon},$$

so the producer's per-period profit, written as a function of the capital stock, is

$$\pi_{jt}^s = M_t\, r_{jt}^s\, s_{ijt} - I_{jt}^s = M_t^{\rho}\, r_{jt}^s\, s_{jt} - I_{jt}^s = M_t^{\rho\left(1-\frac{1}{\varepsilon}\right)+\frac{1}{\varepsilon}}\, \tilde{A}_t^{\frac{1}{\varepsilon}}\, s_{jt}^{1-\frac{1}{\varepsilon}} - I_{jt}^s.$$

Taking M_t and $\tilde{A}_t$ as given, the producer chooses the sequences of investment I_{jt}^s and software stock $s_{j,t+1}$ to solve

$$\max_{\{I_{jt}^s,\, s_{j,t+1}\}} \sum_{t=0}^{\infty} \Lambda_t\, \pi_{jt}^s \qquad \text{s.t.} \qquad s_{j,t+1} = (1-\delta_s) s_{jt} + q_t^s I_{jt}^s,$$

where Λ_t is the household's discount factor. Substituting $I_{jt}^s = \left[s_{j,t+1} - (1-\delta_s) s_{jt}\right] / q_t^s$, the first-order condition for $s_{j,t+1}$ is

$$\frac{\Lambda_t}{q_t^s} = \Lambda_{t+1} \left[\frac{\varepsilon - 1}{\varepsilon} M_{t+1}^{\rho\left(1-\frac{1}{\varepsilon}\right)+\frac{1}{\varepsilon}}\, \tilde{A}_{t+1}^{\frac{1}{\varepsilon}}\, s_{j,t+1}^{-\frac{1}{\varepsilon}} + \frac{1-\delta_s}{q_{t+1}^s} \right].$$

The left-hand side is the cost of one more unit of software capital in $t+1$, valued in period-t terms. The right-hand side is its discounted benefit: the marginal revenue of the stock in $t+1$ plus the replacement value of the undepreciated part. By the inverse demand, the marginal revenue equals $\frac{\varepsilon-1}{\varepsilon} M_{t+1}^{\rho} r_{j,t+1}^s$. The producer's Euler equation is therefore

$$\frac{1}{q_t^s} = \frac{\Lambda_{t+1}}{\Lambda_t} \left[\frac{\varepsilon - 1}{\varepsilon} M_{t+1}^{\rho}\, r_{j,t+1}^s + \frac{1-\delta_s}{q_{t+1}^s} \right]. \tag{A.7}$$

In steady state, $\Lambda_{t+1}/\Lambda_t = \beta$ and $q^s_t = q^s$, so (A.7) gives

$$r^s = \frac{\varepsilon}{\varepsilon - 1}\frac{\kappa}{M^\rho}, \qquad \kappa \equiv \frac{1}{q^s}\left(\frac{1}{\beta} - 1 + \delta_s\right). \tag{A.8}$$

The rental rate is a markup over the user cost of software capital, κ, scaled down by M^ρ. The user cost of a unit of software capital is spread across all M firms that use it, and the specificity parameter ρ governs how far it can be spread: when $\rho = 0$, software is fully specific and each firm bears the full user cost; when $\rho = 1$, software is fully shareable and the user cost is split equally among the M firms. Steady-state investment is $I^s = \delta_s S/q^s$.

Household. The household solves

$$\max_{\{C_t, I^k_t\}} \sum_{t=0}^{\infty} \beta^t \log C_t \qquad \text{s.t.} \qquad C_t + I^k_t = w_t L_t + r^k_t K_t + \Pi^s_t, \qquad K_{t+1} = (1-\delta_k)K_t + q^k_t I^k_t,$$

where Π^s_t is the profit of software producers. The Euler equation is

$$\frac{1}{q^k_t} = \beta \frac{C_t}{C_{t+1}}\left(r^k_{t+1} + \frac{1-\delta_k}{q^k_{t+1}}\right),$$

which also gives the software producers' discount factor, $\Lambda_{t+1}/\Lambda_t = \beta C_t/C_{t+1}$. In steady state the rental rate of capital depends only on parameters:

$$r^k = \frac{1}{q^k}\left(\frac{1}{\beta} - 1 + \delta_k\right). \tag{A.9}$$

Steady-state investment is $I^k = \delta_k K/q^k$.

Free entry. Firms enter until operating profit (A.4) covers the fixed cost f_c, which is paid in units of the final good: $(1-\eta)y_i = f_c$. Since $Y = My_i$,

$$M = \frac{1-\eta}{f_c} Y. \tag{A.10}$$

Substituting into (A.8), the software rental rate falls with aggregate output:

$$r^s = \frac{\varepsilon}{\varepsilon - 1}\kappa\left(\frac{1-\eta}{f_c} Y\right)^{-\rho}.$$

Labor market clearing. The labor market clears when

$$\int_0^M l_i \, di = M l_i = L.$$

Combining with the final-good firm's first-order condition for labor in (A.2), $w l_i = \eta \alpha_l y_i$, and using $Y = M y_i$, gives the wage,

$$w = \eta \alpha_l \frac{Y}{L}.$$

Input ratios. By the relationship between the software stock and services in (A.6), the software first-order condition in (A.5), and the rental rate in (A.8), the software stock is proportional to output:

$$S = \frac{X}{M^\rho} = \frac{\eta \alpha_s Y}{r^s M^\rho} = \frac{\varepsilon - 1}{\varepsilon} \frac{\eta \alpha_s}{\kappa} Y = q^s \frac{\varepsilon - 1}{\varepsilon} \frac{\eta \alpha_s}{\frac{1}{\beta} - 1 + \delta_s} Y.$$

Note that the M^ρ terms cancel. Sharing software raises the services provided by each unit of the stock by the factor M^ρ and lowers the rental rate by the same factor, so revenue per unit of stock, $M^\rho r^s$, does not depend on M.

Therefore, the mass of firms, the capital stock, and the software stock are all proportional to output:

$$M = \mu_m Y, \qquad K = \mu_k Y, \qquad S = q^s \mu_s Y, \tag{A.11}$$

with

$$\mu_m \equiv \frac{1 - \eta}{f_c}, \qquad \mu_k \equiv \frac{\eta \alpha_k q^k}{\frac{1}{\beta} - 1 + \delta_k}, \qquad \mu_s \equiv \frac{\varepsilon - 1}{\varepsilon} \frac{\eta \alpha_s}{\frac{1}{\beta} - 1 + \delta_s}.$$

Here μ_m follows from free entry (A.10), μ_k from the capital condition in (A.5) and the rental rate of capital (A.9), and μ_s from the derivation of S above.

Steady-state output. Substituting (A.11) into the aggregate production function (A.1),

$$Y = z (\mu_m Y)^\psi \left[(\mu_k Y)^{\alpha_k} (q^s \mu_s Y)^{\alpha_s} L^{\alpha_l} \right]^\eta.$$

Collecting the Y terms and taking logs gives steady-state output,

$$\log Y = \frac{1}{\eta (\alpha_l - \alpha_s \rho)} \Big[\log z + \psi \log \mu_m + \eta \alpha_k \log \mu_k + \eta \alpha_s \log (q^s \mu_s) + \eta \alpha_l \log L \Big]. \tag{A.12}$$

We assume $\alpha_l > \alpha_s \rho$, so the steady state is well defined. Equivalently, the inputs that scale with output, M, K, and S, have combined returns $\psi + \eta(\alpha_k + \alpha_s) < 1$ in (A.1).

A.3 Comparative Statics

Since μ_m, μ_k, and μ_s depend on neither L nor q^s, both elasticities follow from differentiating (A.12).

Labor force. Differentiating (A.12) with respect to $\log L$ gives

$$\frac{\partial \log Y}{\partial \log L} = \frac{\eta \alpha_l}{\eta (\alpha_l - \alpha_s \rho)} = \frac{\alpha_l}{\alpha_l - \alpha_s \rho}.$$

The elasticity of output per worker with respect to labor is therefore

$$\frac{\partial \log(Y/L)}{\partial \log L} = \frac{\partial \log Y}{\partial \log L} - 1 = \frac{\alpha_l}{\alpha_l - \alpha_s \rho} - 1 = \frac{\alpha_s \rho}{\alpha_l - \alpha_s \rho} = \frac{\alpha_s \rho}{1 - \alpha_k - \alpha_s - \alpha_s \rho}. \tag{A.13}$$

Software productivity. Differentiating (A.12) with respect to $\log q^s$ gives

$$\frac{\partial \log Y}{\partial \log q^s} = \frac{\eta \alpha_s}{\eta (\alpha_l - \alpha_s \rho)} = \frac{\alpha_s}{\alpha_l - \alpha_s \rho} = \frac{\alpha_s}{1 - \alpha_k - \alpha_s - \alpha_s \rho}. \tag{A.14}$$

Since L is fixed, this is also the elasticity of output per worker, $\partial \log(Y/L)/\partial \log q^s$.

A.4 Efficiency

We compare the steady state of the decentralized equilibrium with that of the planner's allocation. We first solve the planner's problem and then derive the gaps in inputs and output between the two allocations.

Planner's problem. The aggregate production function (A.1) uses only technology, symmetry, and the resource constraint for software, so it also describes the production possibilities available to a social planner. The planner chooses consumption, the mass of firms, and the two capital stocks to solve

$$\max_{\{C_t, M_t, K_{t+1}, S_{t+1}\}} \sum_{t=0}^{\infty} \beta^t \log C_t$$

subject to the final-good resource constraint

$$C_t + \frac{K_{t+1} - (1-\delta_k) K_t}{q_t^k} + \frac{S_{t+1} - (1-\delta_s) S_t}{q_t^s} + M_t f_c = Y_t, \qquad Y_t = z M_t^{1-\eta+\eta\alpha_s\rho} \left(K_t^{\alpha_k} S_t^{\alpha_s} L^{\alpha_l}\right)^{\eta},$$

given K_0 and S_0.

First-order conditions. Substituting C_t from the resource constraint, the first-order conditions for M_t, K_{t+1}, and S_{t+1} are

$$\begin{aligned} f_c &= (1-\eta+\eta\alpha_s\rho)\,\frac{Y_t}{M_t}, \\ \frac{1}{q_t^k} &= \beta\frac{C_t}{C_{t+1}}\left[\eta\alpha_k\frac{Y_{t+1}}{K_{t+1}}+\frac{1-\delta_k}{q_{t+1}^k}\right], \\ \frac{1}{q_t^s} &= \beta\frac{C_t}{C_{t+1}}\left[\eta\alpha_s\frac{Y_{t+1}}{S_{t+1}}+\frac{1-\delta_s}{q_{t+1}^s}\right]. \end{aligned}$$

The planner values an additional firm at its full marginal product, which includes the gain from spreading the software stock over more users, $\eta\alpha_s\rho\,Y/M$. A private entrant earns only $(1-\eta)y_i = (1-\eta)Y/M$. Similarly, the planner values an additional unit of software at its marginal product $\eta\alpha_s Y/S$, while a software producer earns only the fraction $\frac{\varepsilon-1}{\varepsilon}$ of it because of the markup.

Steady state. In steady state, $C_{t+1}=C_t$, $q_t^k=q^k$, and $q_t^s=q^s$, so the planner's allocation satisfies

$$M^P=\mu_m^P Y^P, \qquad K^P=\mu_k^P Y^P, \qquad S^P=q^s\mu_s^P Y^P,$$

with

$$\mu_m^P\equiv\frac{1-\eta+\eta\alpha_s\rho}{f_c}, \qquad \mu_k^P\equiv\frac{\eta\alpha_k q^k}{\frac{1}{\beta}-1+\delta_k}, \qquad \mu_s^P\equiv\frac{\eta\alpha_s}{\frac{1}{\beta}-1+\delta_s}.$$

Sources of inefficiency. Since each input is its ratio times output, comparing the planner's ratios with (A.11) gives the gaps in inputs between the two allocations, where the output gap Y^D/Y^P is given by (A.17) below. First, final-good firms do not internalize the benefit of their entry for software sharing, so there is too little entry relative to the planner's allocation:

$$\frac{M^D}{M^P}=\frac{\mu_m}{\mu_m^P}\frac{Y^D}{Y^P}=\frac{1-\eta}{1-\eta+\eta\alpha_s\rho}\frac{Y^D}{Y^P}. \tag{A.15}$$

The term $\eta\alpha_s\rho$ in the denominator is the social benefit of entry from expanding the software user base. Second, software producers charge a markup over the marginal cost of providing software services, so there is too little investment in software capital:

$$\frac{S^D}{S^P}=\frac{\mu_s}{\mu_s^P}\frac{Y^D}{Y^P}=\frac{\varepsilon-1}{\varepsilon}\frac{Y^D}{Y^P}. \tag{A.16}$$

Traditional capital is not distorted directly: its ratio to output is the same in both allocations, so $K^D/K^P=Y^D/Y^P$ and it falls short only in proportion to output.

Output gap. Because both allocations satisfy (A.1) with inputs proportional to output, (A.12) holds for each allocation with its own ratios. Subtracting the planner's version from the decentralized one, all terms cancel except those involving the ratios for firms and software in (A.15) and (A.16), and the output gap is

$$\frac{Y^D}{Y^P} = \left[\left(\frac{1-\eta}{1-\eta+\eta\alpha_s\rho} \right)^{\frac{1-\eta+\eta\alpha_s\rho}{\eta}} \left(\frac{\varepsilon-1}{\varepsilon} \right)^{\alpha_s} \right]^{\frac{1}{1-\alpha_k-\alpha_s-\alpha_s\rho}} < 1. \tag{A.17}$$

Table 9: The Economic Environment

$y_{i,t} = z_t \left(k_{i,t}^{\alpha_k} s_{i,t}^{\alpha_s} l_{i,t}^{\alpha_l}\right)^{\eta}$, where $s_{i,t} = \left(\int_0^1 s_{ij,t}^{\frac{\varepsilon-1}{\varepsilon}} dj\right)^{\frac{\varepsilon}{\varepsilon-1}}$ and $\alpha_k + \alpha_s + \alpha_l = 1, \eta < 1$	Final-good firm production
$\pi_{i,t} = y_{i,t} - w_t l_{i,t} - r_t^k k_{i,t} - \int_0^1 r_{j,t}^s s_{ij,t} dj$	Final-good firm per-period gross profit (max)
$s_{ij,t} = \tilde{A}_{i,t} (r_{j,t}^s)^{-\varepsilon}$ where $\tilde{A}_{i,t} \equiv \eta \alpha_s y_{i,t} (R_t^s)^{\varepsilon-1}$ and $R_t^s \equiv \left(\int_0^1 (r_{j,t}^s)^{1-\varepsilon} dj\right)^{\frac{1}{1-\varepsilon}}$	Demand for software variety j by firm i
$\pi_{i,t} = (1-\eta) y_{i,t} = f_c$	Free-entry condition for final-good firms
$\pi_{j,t}^s = \int_0^{M_t} r_{j,t}^s s_{ij,t} di - I_{j,t}^s$	Software producer per-period profit (max)
$\max_{\{I_{j,t}^s, s_{j,t+1}, r_{j,t}^s\}} \sum_{t=0}^{\infty} \Lambda_t \pi_{j,t}^s$	Software producer dynamic profit
$s_{j,t+1} = (1-\delta_s) s_{j,t} + q_t^s I_{j,t}^s$	Software capital accumulation
$s_{j,t} \geq \left(\int_0^{M_t} s_{ij,t}^{\frac{1}{1-\rho}} di\right)^{1-\rho}, \forall j$	Software capital resource constraint
$\sum_{t=0}^{\infty} \beta^t \log C_t$	Household utility
$C_t + I_t^k = w_t L_t + r_t^k K_t + \Pi_t^s$ where $\Pi_t^s = \int_0^1 \pi_{j,t}^s dj$	Household budget constraint
$K_{t+1} = (1-\delta_k) K_t + q_t^k I_t^k$	Physical capital accumulation
$L_t = \int_0^{M_t} l_{i,t} di$	Labor resource constraint
$K_t = \int_0^{M_t} k_{i,t} di$	Physical capital resource constraint
$Y_t = C_t + I_t^k + I_t^s + M_t f_c$, where $I_t^s = \int_0^1 I_{j,t}^s dj$	Final good resource constraint
$S_t = \int_0^1 s_{j,t} dj$	Aggregate software capital
$Y_t = M_t y_{i,t}$	Aggregate output
$X_t = M_t s_{i,t} = S_t M_t^{\rho}$	Aggregate software services

Notes: This table summarizes the economic environment of the simple model in Section 2.

Table 10: The Economic Environment: Quantitative Model

$y_{it}^n = A_t z_t^n \left(k_{it}^{\beta_k} l_{it}^{\beta_l}\right)^{\eta}, \qquad \beta_k + \beta_l = 1,$	
$y_{it}^a = A_t z_t^a \left(k_{it}^{\alpha_k} s_{it}^{\alpha_s} l_{it}^{\alpha_l}\right)^{\eta}, \quad \alpha_k + \alpha_s + \alpha_l = 1, \quad \eta < 1,$	Final-good technologies
$s_{it} = \left(\int_0^1 s_{ijt}^{\frac{\varepsilon-1}{\varepsilon}} dj\right)^{\frac{\varepsilon}{\varepsilon-1}}$	
$\pi_t^n = y_{it}^n - r_t^k k_{it}^n - w_t l_{it}^n,$	
$\pi_t^a = y_{it}^a - r_t^k k_{it}^a - w_t l_{it}^a - \int_0^1 r_{jt}^s s_{ijt}\, dj$	Final-good firm gross profits
$s_{ijt} = \widetilde{A}_{it}(r_{jt}^s)^{-\varepsilon}, \qquad \widetilde{A}_{it} \equiv \eta\, \alpha_s y_{it}^a (R_t^s)^{\varepsilon-1},$	
$R_t^s \equiv \left(\int_0^1 (r_{jt}^s)^{1-\varepsilon} dj\right)^{\frac{1}{1-\varepsilon}}$	Demand for software variety j
$\Delta\pi_t \equiv \pi_t^a - \pi_t^n, \qquad \text{adopt iff } \Delta\pi_t \geq w_t f_i^a, \quad f_i^a \sim G(f^a),$	
$P_t^a = G\left(\frac{\Delta\pi_t}{w_t}\right), \qquad M_t^a = M_t P_t^a, \qquad M_t^n = M_t(1-P_t^a)$	Adoption and firm masses
$\mathbb{E}_i[\max\{\pi_t^n, \pi_t^a - w_t f_i^a\}] = w_t f_c$	Free-entry condition
$\pi_{jt}^s = \int_0^{M_t^a} r_{jt}^s s_{ijt}\, di - I_{jt}^s$	Software producer per-period profit
$\max_{\{I_{jt}^s, s_{j,t+1}, r_{jt}^s\}_{t\geq 0}} \sum_{t=0}^{\infty} \Lambda_t \pi_{jt}^s$	Software producer dynamic problem
$s_{j,t+1} = (1-\delta_s) s_{jt} + q_t^s I_{jt}^s$	Software capital accumulation
$s_{jt} \geq \frac{s_{ijt}}{l_{it}^a}(L_t^a)^{1-\rho}, \qquad L_t^a = \int_0^{M_t^a} l_{it}^a\, di, \qquad \rho_l = \rho_m = \rho$	Software capital resource constraint
$\max_{\{C_t, I_t^k\}_{t\geq 0}} \sum_{t=0}^{\infty} \beta^t \log C_t$	Household utility
$C_t + I_t^k = w_t L_t + r_t^k K_t + \Pi_t^s, \qquad \Pi_t^s \equiv \int_0^1 \pi_{jt}^s\, dj$	Household budget constraint
$K_{t+1} = (1-\delta_k) K_t + q_t^k I_t^k$	Physical capital accumulation
$L_t = L_t^a + L_t^n + M_t^a \mathbb{E}_i\left[f_i^a \mid f_i^a \leq \frac{\Delta\pi_t}{w_t}\right] + M_t f_c,$	Labor resource constraint
$L_t^h = M_t^h l_{it}^h, \qquad h \in \{a, n\}$	
$K_t = K_t^a + K_t^n, \qquad K_t^h = M_t^h k_{it}^h, \quad h \in \{a, n\}$	Physical capital resource constraint
$Y_t = C_t + I_t^k + I_t^s, \qquad I_t^s \equiv \int_0^1 I_{jt}^s\, dj$	Final-good resource constraint
$S_t = \int_0^1 s_{jt}\, dj$	Aggregate software capital
$Y_t = Y_t^a + Y_t^n = M_t^a y_{it}^a + M_t^n y_{it}^n$	Aggregate output
$X_t = M_t^a s_{it} = S_t (L_t^a)^{\rho}$	Aggregate software services

Notes: This table summarizes the economic environment of the quantitative model in Section 4.

B Growth Accounting Appendix

Table 11: Calibration with different values of ρ

(a) $\rho = 0.5$

Param.	Description	Value	Source	Data	Model
β	Discount factor	0.960	–	–	–
δ_k	Depreciation k	0.132	BEA	0.132	0.132
δ_s	Depreciation sw	0.330	BEA	0.330	0.330
η	DRTS	0.850	Profit share, Barkai	0.150	0.150
β_l	Non-adopter labor exponent	0.561	BEA labor and software shares	0.605	0.605
α_l	Adopter labor exponent	0.510	$\beta_l - \alpha_s$	–	–
β_k, α_k	Capital exponent	0.439	$1 - \beta_l$	–	–
α_s	Software share adopters	0.051	BEA agg. sw share	0.022	0.022
z_a	Productivity adopters	1.175	Rel. adopter emp., ACES	1.400	1.400
f_c	Fixed cost	0.003	Production/software firm count, BDS	81.132	81.132
q_k	ISTC	1.467	K/Y, BEA–BLS	3.160	3.160
q_s	SW-STC	0.897	X/Y, BEA–BLS	0.024	0.024
μ_a	Mean of log adoption cost	-6.324	Adoption share, ACES	0.400	0.400
σ_a	Sd of log adoption cost	0.896	SW-share–r^s log slope	-0.731	-0.731

Notes: z_n, L, and the mass of software-producing firms are normalized to one.

(b) $\rho = 1.0$

Param.	Description	Value	Source	Data	Model
β	Discount factor	0.960	–	–	–
δ_k	Depreciation k	0.132	BEA	0.132	0.132
δ_s	Depreciation sw	0.330	BEA	0.330	0.330
η	DRTS	0.850	Profit share, Barkai	0.150	0.150
β_l	Non-adopter labor exponent	0.561	BEA labor and software shares	0.605	0.605
α_l	Adopter labor exponent	0.510	$\beta_l - \alpha_s$	–	–
β_k, α_k	Capital exponent	0.439	$1 - \beta_l$	–	–
α_s	Software share adopters	0.051	BEA agg. sw share	0.022	0.022
z_a	Productivity adopters	1.175	Rel. adopter emp., ACES	1.400	1.400
f_c	Fixed cost	0.003	Production/software firm count, BDS	81.132	81.132
q_k	ISTC	1.467	K/Y, BEA–BLS	3.160	3.160
q_s	SW-STC	1.480	X/Y, BEA–BLS	0.024	0.024
μ_a	Mean of log adoption cost	-6.324	Adoption share, ACES	0.400	0.400
σ_a	Sd of log adoption cost	0.896	SW-share–r^s log slope	-0.731	-0.731

Notes: z_n, L, and the mass of software-producing firms are normalized to one.

B.1 The growth accounting framework

Output in the model is given by aggregate production:

$$\begin{aligned} Y_t &= Y_t^a + Y_t^n, \\ Y_t^a &= (M_t^a)^{1-\eta} A_t z_t^a \left[(K_t^a)^{\alpha_k} X_t^{\alpha_s} (L_t^a)^{\alpha_l} \right]^{\eta}, \qquad X_t = S_t (L_t^a)^{\rho}, \\ Y_t^n &= (M_t^n)^{1-\eta} A_t z_t^n \left[(K_t^n)^{\beta_k} (L_t^n)^{\beta_l} \right]^{\eta}. \end{aligned}$$

The exact decomposition of labor-productivity growth

$$\Delta \log(Y/L) = C_{\text{TFP}} + C_{K,\text{deep}} + \underbrace{C_S + C_{\text{reuse}}}_{C_X}$$

where

$$\begin{aligned} C_{\text{TFP}} &= C_A + C_z + C_M + C_{K,\text{realloc}} + C_{L,\text{realloc}}, \\ C_S &= \Omega_a \eta \alpha_s \Delta \log \left(\frac{S}{L} \right), \\ C_{\text{reuse}} &= \Omega_a \eta \alpha_s \rho \Delta \log L^a, \\ C_{K,\text{deep}} &= (\Omega_a \eta \alpha_k + \Omega_n \eta \beta_k) \Delta \log \left(\frac{K}{L} \right). \end{aligned}$$

and Ω_x is approximately the Tornqvist weights:

$$\Omega_a = \frac{\mathscr{L}\left(\widetilde{Y}_1^a, \widetilde{Y}_0^a\right)}{\mathscr{L}\left(\widetilde{Y}_1, \widetilde{Y}_0\right)}, \qquad \Omega_n = \frac{\mathscr{L}\left(\widetilde{Y}_1^n, \widetilde{Y}_0^n\right)}{\mathscr{L}\left(\widetilde{Y}_1, \widetilde{Y}_0\right)}.$$

$\mathscr{L}$ is the logarithmic mean:

$$\mathscr{L}(u_1, u_0) = \frac{u_1 - u_0}{\log u_1 - \log u_0}, \qquad \mathscr{L}(u, u) = u.$$

The TFP term is given by:

$$C_{\text{TFP}} = C_A + C_z + C_M + C_{K,\text{realloc}} + C_{L,\text{realloc}}.$$

$$C_A = (\Omega_a + \Omega_n)\Delta \log A,$$

$$C_z = \Omega_a \Delta \log z^a + \Omega_n \Delta \log z^n,$$

$$C_{K,\text{realloc}} = \Omega_a \eta \alpha_k \Delta \log \left(\frac{K^a}{K} \right) + \Omega_n \eta \beta_k \Delta \log \left(\frac{K^n}{K} \right),$$

$$C_{L,\text{realloc}} = \Omega_a \eta \alpha_l \Delta \log \left(\frac{L^a}{L} \right) + \Omega_n \eta \beta_l \Delta \log \left(\frac{L^n}{L} \right),$$

$$C_M = \Omega_a (1-\eta) \Delta \log \left(\frac{M^a}{L} \right) + \Omega_n (1-\eta) \Delta \log \left(\frac{M^n}{L} \right).$$