PROGRESS ON
Endogenous Production Networks under Supply Chain Uncertainty

Jinhua Wu

August 3, 2024

Contents

1 Some Notes on Introdution Part

1.1 Definitions

Production Network A production network refers to a complex system of interconnected entities and processes involved in the production and distribution of goods and services. This network includes suppliers, manufacturers, distributors, retailers, and end customers, all working together to ensure the efficient flow of products from raw materials to finished goods. The production network encompasses various stages such as procurement, manufacturing, logistics, and sales, each playing a crucial role in maintaining the overall efficiency and effectiveness of the production process. Effective management of a production network can lead to improved productivity, cost savings, and competitive advantage.

Domar Weights Domar weights, named after the economist Evsey Domar, are used to measure the contribution of each sector to the overall economy. Specifically, in a production network, the Domar weight of a sector is the ratio of that sector's output to the total GDP. This weight reflects the relative importance of a sector in the economy, considering both direct and indirect contributions through the production network. The concept is crucial in understanding how shocks to different sectors can propagate through the economy and affect overall productivity and welfare.

Rish-averse representative household A risk-averse representative household is a theoretical construct used in economic models to represent the behavior of a typical household that prefers to avoid risk. This household supplies a fixed amount of labor and makes consumption decisions to maximize its utility, which depends on the consumption of various goods. The utility function used in the model typically exhibits constant relative risk aversion (CRRA), meaning the household's aversion to risk remains constant regardless of its wealth level. The household makes consumption decisions after uncertainty in the economy is resolved, facing a budget constraint based on the prices of goods and the household's income. The risk aversion parameter (ρ) in the utility function quantifies how much the household dislikes risk: a higher (ρ) indicates greater risk aversion. The household's decisions influence the production network because firms take into account the household's preferences and risk aversion when making their own production and pricing decisions.

TFP Process The TFP process refers to the Total Factor Productivity process, which is a crucial component in understanding economic growth and production efficiency. TFP measures the efficiency with which labor and capital are used together in the production process. The TFP process involves both the endogenous and exogenous factors that affect productivity in different sectors of the economy.

Risk exposure Risk exposure refers to the extent to which an entity (such as a firm, household, or economy) is vulnerable to various types of risks that can affect its performance or stability. In an economic context, risk exposure often involves uncertainties related to price fluctuations, supply chain disruptions, productivity shocks, and other external factors that can impact costs, revenues, and overall economic welfare.

Variance of Unit Costs: Firms prefer inputs with stable prices and avoid techniques relying on inputs with positively correlated prices. This helps in diversifying risk and minimizing cost volatility.

Correlation with Productivity Shocks: Firms prefer inputs whose prices are positively correlated with their productivity shocks. This means that during a negative shock, input prices are likely to be low, reducing expected cost increases.

Risk-Adjusted Prices: Firms' technique choices are influenced by risk-adjusted prices, which account for the expected price of inputs and their covariance with the stochastic discount factor. Goods that are cheaper when aggregate consumption is low are particularly attractive.

Impact on Supply Chain: Higher supplier volatility increases the likelihood of link destruction in supply relationships. Firms tend to move away from riskier suppliers to ensure stability.

Hulten's Theorem Hulten's theorem, named after economist Charles R. Hulten, is a fundamental result in the field of growth accounting and productivity analysis. The theorem states that the aggregate output (GDP) of an economy is a weighted sum of the outputs of its individual sectors, with the weights being the sectoral shares in total output. In simple terms, it implies that the proportional change in aggregate output is equal to the weighted sum of the proportional changes in the output of individual sectors.

Mathematically, if ΔY represents the change in aggregate output and Δy_i represents the change in the output of sector i, Hulten's theorem can be expressed as:

$$\Delta Y = \sum_i w_i \Delta y_i$$

where w_i is the Domar weight of sector i, reflecting its importance in the overall economy. It simplifies the analysis of how shocks to individual sectors affect the whole economy. It assumes a fixed production network, meaning the input-output relationships between sectors do not change in response to the shocks.

Alternative Economy An alternative economy refers to an economic system or a set of practices that differ from the traditional market-driven economy. It encompasses a wide range of economic models and activities that prioritize social, environmental, and ethical considerations over profit maximization. These alternative economic systems often emphasize community-oriented, cooperative, and sustainable practices.

In the context of the provided document, alternative economies are used as benchmarks to evaluate the impact of various factors such as uncertainty on the production network and macroeconomic aggregates. Specifically, the document compares the baseline economy to alternative economies where firms are either unconcerned about risk when making sourcing decisions or have perfect foresight of productivity shocks. These comparisons help isolate the impact of uncertainty on the production network and its subsequent effect on GDP and welfare.

Multi-sector economy A multi-sector economy refers to an economic model that includes multiple sectors or industries, each producing different goods or services. This approach allows for a more detailed and realistic analysis of the economy by capturing the interactions and dependencies between various sectors. In a multi-sector economy, each sector may have its own production function, input requirements, and productivity shocks, and the outputs of some sectors serve as inputs for others, creating a complex network of interconnections.

Productivity shifter the productivity shifter is a function that represents how effectively a sector combines its inputs to produce output. It reflects the total factor productivity (TFP) of the sector, which varies depending on the chosen production technique α_i. This shifter function is crucial in determining the productivity level of a sector and is influenced by the allocation of input shares among different suppliers.

Aggregate Risk refers to the overall level of risk that affects the entire economy or a significant portion of it. It encompasses the uncertainties and potential fluctuations in economic variables that can impact multiple sectors simultaneously. Unlike idiosyncratic risk, which affects only individual firms or sectors, aggregate risk involves macroeconomic factors that can influence the entire economic system.

Pareto Efficient Allocations A Pareto efficient allocation is a state of resource distribution where it is impossible to make any individual better off without making at least one individual worse off. In other words, an allocation is Pareto efficient if no further reallocation can improve someone's situation without harming another person's situation. This concept is named after the Italian economist Vilfredo Pareto.

1.2 Summary for innovations

Modeling Supply Chain Uncertainty The authors construct a model of endogenous network formation to investigate how firms' decisions to mitigate supply chain risks affect the production network and macroeconomic aggregates. This model builds on and extends the work of Acemoglu and Azar (2020).

Focus on Uncertainty Unlike previous models that assume firms know the realization of shocks when choosing production techniques, this model incorporates uncertainty and beliefs about future productivity shocks into the decision-making process. This change allows the model to capture the impact of uncertainty on the structure of the production network.

Technique Choice and Production Network The model allows firms to choose production techniques that specify which intermediate inputs to use and how to combine them. These techniques can vary in terms of productivity, and firms can adjust the importance of suppliers or drop them altogether. This flexibility captures adjustments in the production network along both intensive and extensive margins.

Risk-Adjusted Prices Firms in the model choose techniques by considering risk-adjusted prices, reflecting the risk attitude of the representative household. This approach shows how aggregate risk and firms' sourcing decisions interact to shape the production network.

Empirical Relevance The authors provide a basic calibration of the model using U.S. data to evaluate the importance of these mechanisms. They also highlight the model's ability to predict that increased uncertainty leads firms to prefer more stable suppliers, which reduces macroeconomic volatility but also lowers aggregate output.

Comparative Analysis with Alternative Economies The paper compares the baseline economy with alternative economies where firms either do not consider risk in their sourcing decisions or have perfect foresight of productivity shocks. This comparison helps to isolate the impact of uncertainty on the production network and macroeconomic outcomes.

2 Model

Notations and Symbols

Notations	Meanings
ρ	The utility function quantifies how much the household dislikes risk
$i \in \{1, \cdots, n\}$	n sectors
$\mathcal{A}_i$	The representative firm in sector $i$has access to a set of production techniques
$\alpha_i = (\alpha_{i1}, \cdots, \alpha_{in}) \in \mathcal{A}_i$	Inputs used in production and combined in production
$A_i(\alpha_i)$	a productivity shitfer
L_i	Labor
$X_i = (X_{i1}, \cdots, X_{in})$	A vector of intermediate inputs
ε_i	Stochastic component of sector i's total factor productivity
$\varepsilon \sim \mathcal{N}(\mu, \Sigma)$	Collect the previous shock $\varepsilon = (\varepsilon_1, \cdots, \varepsilon_n)$
$\zeta(\alpha_i)$	A normalization to simplify future expressions
$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$	Cartesian product
$C = (C_1, \cdots, C_n)$	consumption vector
$u(\cdot)$	CRRA with a coefficient of relative risk aversion $\rho \geqslant 1$
P_i	the price of good i
Λ	Stochastic discount factor
$\overline{P}$	Price index
β	consumption shares
$K_i(\alpha_i, P)$	The unit cost of production
Q_i	the equilibrium demand for good i
$\mathcal{L}(\alpha) = (I - \alpha)^{-1}$	The Leontief inverse
ω_i	Domar weight of sector i
α_i^*	a technique to maximize expected discounted profits
$\lambda(\alpha^*)$	stochastic discount factor
$k_i(\alpha_i, \alpha^*)$	The log of unit cost
$\mathcal{R}(\alpha^*)$	The vector of equilibirum risk-adjusted price

2.1 Firms and production functions

The corresponding production function

$$F(\alpha_i, L_i, X_i) = e^{\varepsilon_i} A_i(\alpha_i)\zeta(\alpha_i) L_i^{1-\sum_{j=1}^{n}\alpha_{ij}} \prod_{j=1}^{n} X_{ij}^{\alpha_{ij}} \tag{1}$$

where L_i is labor and $X_i = (X_{i1}, \cdots, X_{in})$ is a vector of intermediate inputs. The term ε_i is the stochastic component of sector i's total factor productivity. Finally, $\zeta(\alpha_i)$ is a normalization to simplify future expressions.

Set of feasible production techniques

$$\mathcal{A}_i = \left\{ \alpha_i \in [0,1]^n : \sum_{j=1}^{n} \alpha_{ij} \leqslant \bar{\alpha}_i \right\}$$

where $0 < 1 - \bar{\alpha}_i < 1$ provides a lower bound on the share of labor in the production of good i.

Assumption 1. $A_i(\alpha_i)$ is smooth and strictly log-concave.

For each sector i, there is a unique vector of ideal input shares $\alpha_i^{\circ} \in \mathcal{A}_i$ that maximize A_i and that represents the most productive way to combine intermediate inputs to produce good i. **We normalize $A_i(\alpha_i^{\circ}) = 1$ for all i.**

Example One example of a function $A_i(\alpha_i)$ that satisfies Assumption 1 is the quadratic form

$$\log A_i(\alpha_i) = \frac{1}{2}(\alpha_i - \alpha_i^{\circ})^T \bar{H}_i (\alpha_i - \alpha_i^{\circ}) \tag{2}$$

where $\bar{H}_i$ is a negative-definite matrix that is also the Hessian of $\log A_i$.

2.2 Household preferences

CRRA A risk-averse representative household supplies one unit of labor in elastically and chooses aconsumption vector $C = (C_1, \cdots, C_n)$ to maximize

$$u\left(\left(\frac{C_1}{\beta_1}\right)^{\beta_1} \cdots \left(\frac{C_n}{\beta_n}\right)^{\beta_n}\right) \tag{3}$$

where $\beta_i > 0$ for all i and $\sum_{i=1}^{n} \beta_i = 1$. We refer to $Y = \prod_{i=1}^{n} (\beta_i^{-1} C_i)^{\beta_i}$ as aggregate consumption or, equivalently in this setting, GDP. The utility function $u(\cdot)$ is CRRA[1] with a coefficient of relative risk aversion $\rho \geqslant 1$. The household makes consumption decisions after uncertainty is resolved and so in each state of the world it faces the budget constraint

$$\sum_{i=1}^{n} P_i C_i \leqslant 1 \tag{4}$$

where P_i is the price of good i, and the wage is used as the numeraire.

Stochastic discount factor Firms are owned by the representative household and maximize expected profits discounted by the household's stochastic discount factor

$$\Lambda = u'(Y)/\overline{P} \tag{5}$$

wbere $\overline{P} = \prod_{i=1}^{n} P_i^{\beta_i}$ is the price index.

Log GDP From the optimization problem of the household it is straightforward to show that

$$y = -\beta^T p \tag{6}$$

where $y = \log Y$, $p = (\log P_1, \cdots, \log P_n)$ and $\beta = (\beta_1, \cdots, \beta_n)$. Log GDP is thus the negative of the sum of log prices weighted by the consumption shares β. Intuitively, as prices decrease relative to wages, the household can purchase more goods, and aggregate consumption increases.

2.3 Unit cost minimization

The second stage problem Under a given technique α_i, the cost minimization problem of a firm in sector i is

$$K_i(\alpha_i, P) = \min_{L_i, X_i} \left(L_i + \sum_{j=1}^{n} P_j X_{ij} \right), \quad \text{subject to } F(\alpha_i, L_i, X_i) \geqslant 1 \tag{7}$$

[1]CRRA stands for Constant Relative Risk Aversion. It is a type of utility function used in economics to describe the behavior of agents who have a consistent attitude towards risk, regardless of their wealth level. The CRRA utility function is commonly used in models of consumer behavior, finance, and macroeconomics because it has several desirable properties, including scalability and tractability.

Thus we construct a Lagrangian Function as:

$$\mathcal{L} = L_i + \sum_{j=1}^{n} P_j X_{ij} + \lambda \left(1 - e^{\varepsilon_i} A_i(\alpha_i)\zeta(\alpha_i) \left(\prod_{j=1}^{n} X_{ij}^{\alpha_{ij}} \right) L_i^{\left(1-\sum\limits_{j=1}^{n} \alpha_{ij}\right)} \right)$$

First-Order Conditions: Taking the first-order conditions with respect to L_i, X_{ij}, and λ, we get:

$$0 = 1 - \left(1 - \sum_{j=1}^{n} \alpha_{ij}\right) e^{\varepsilon_i} \lambda A_i(\alpha_i)\zeta(\alpha_i) \left(\prod_{j=1}^{n} X_{ij}^{\alpha_{ij}} \right) L_i^{\left(-\sum\limits_{j=1}^{n} \alpha_{ij}\right)}$$

$$0 = P_j - \lambda e^{\varepsilon_i} A_i(\alpha_i)\zeta(\alpha_i) L_i^{\left(1-\sum\limits_{j=1}^{n} \alpha_{ij}\right)} \left(\prod_{j=1}^{n} X_{ij}^{\alpha_{ij}} \right) X_{ij}^{-1} \alpha_{ij}$$

Thus we could get the following things:

$$L_i = \left(1 - \sum_{j=1}^{n} \alpha_{ij}\right) \lambda$$

$$X_{ij} = \frac{\lambda \alpha_{ij}}{P_j}$$

Thus we could substitute to the equation and get the following:

$$K_i(\alpha_i, P) = \frac{1}{e^{\varepsilon_i} A_i(\alpha_i)} \prod_{j=1}^{n} P_j^{\alpha_{ij}} \tag{8}$$

2.4 Technique choice

The first stage problem Given an expression for K_i, the first stage of the representative firm's problem is to pick a technique $\alpha_i \in \mathcal{A}_i$ to maximize expected discounted profits, that is,

$$\alpha_i^* \in \arg\max_{\alpha_i \in \mathcal{A}_i} \mathbb{E} \left[\Lambda Q_i (P_i - K_i(\alpha_i, P)) \right] \tag{9}$$

where Q_i is the equilibrium demand for good i, and where the profits in different states of the world are weighted by the household's stochastic discount factor Λ. The representative firm takes P, Q_i and Λ as given, and so the only term in (9) over which it has any control is the unit cost $K_i(\alpha_i, P)$.

2.5 Equilibrium conditions

Competitive Pressure In equilibrium, competitive pressure pushes prices to be equal to unit costs, so that

$$P_i = K_i(\alpha_i, P) \quad \text{for all } i \in \{1, 2, \cdots, n\} \tag{10}$$

Definition 1. An equilibirum is a choice of technique $\alpha^* = (\alpha_1^*, \cdots, \alpha_n^*)$ and a stochastic tuple $(P^*, C^*, L^*, X^*, Q^*)$ such that

1. (Optimal technique choice) For each $i \in \{1, 2, \cdots, n\}$, the technique choice $\alpha_i^* \in \mathcal{A}_i$ solves (9) given price P^*, demand Q_i^* and the stochastic discount factor Λ^* given by (5).

2. (Optimal input choice) For each $i \in \{1, 2, \cdots, n\}$, factor demands per unit of output L_i^*/Q_i^* and X_i^*/Q_i^* are a solution to (7) given price P^* and the chosen technique α_i*.

3. (Consumer maximization) The consumption vector C^* maximizes (3) subject to (4) given prices P^*.

4. (Unit cost pricing) For each $i \in \{1, 2, \cdots, n\}$, P_i^* solves (10) where $K_i(\alpha_i^*, P^*)$ is given by (8).

5. (Market clearning) For each $i \in \{1, 2, \cdots, n\}$,

$$C_i^* + \sum_{j=1}^{n} X_{ji}^* = Q_i^* = F_i(\alpha_i^*, L_i^*, X_i^*), \quad \text{and } \sum_{i=1}^{n} L_i^* = 1 \tag{11}$$

3 Equilibirum prices and GDP in a fixed-network economy

Domar weight We also define the Domar weight ω_i of sector i as the ratio of its sales to nominal GDP, such that

$$\omega_i = \frac{P_i Q_i}{P^T C}$$

Also $\omega^T = \beta^T \mathcal{L}(\alpha) > 0$ in the model.

Lemma 1. Under a given network α, the vector of log prices is given by

$$p(\alpha) = -\mathcal{L}(\alpha)(\varepsilon + a(\alpha)) \tag{12}$$

and log GDP is given by

$$y(a) = \omega(a)^T(\varepsilon + a(\alpha)) \tag{13}$$

where $a(\alpha) = (\log A_i(\alpha_i), \cdots, \log A_n(\alpha_n))$

Proof. Combining the unit cost equation (8) with the equilibrium condition (10) and taking the log we could get

$$p_i = \log P_i = \log K_i(\alpha, P) = -\varepsilon_i - \log A_i(\alpha_i) + \sum_{j=1}^{n} \alpha_{ij} \log P_j = -\varepsilon_i - a_i(\alpha_i) + \sum_{j=1}^{n} \alpha_{ij} p_j$$

where $a_i(\alpha_i) = \log(A_i(\alpha_i))$. This is a system of linear equations whose solution is (12). The log price vector is also normally distributed since it is a linear transformation of normal random variable. Combining with (6) yields (13).

$$y = -\beta^T p = -\beta^T - \mathcal{L}(\alpha)(\varepsilon + a(\alpha)) = \omega^T(\varepsilon + a(\alpha))$$

$\omega^T = \beta^T \mathcal{L}(\alpha)$

$\square$

The first and second moments

$$\mathbb{E}[y(\alpha)] = \omega(a)^T(\mu + a(\alpha)) \quad \mathbb{V}[y(\alpha)] = \omega(a)^T \Sigma \omega(\alpha) \tag{14}$$

Corollary 1. For a fixed production network α, the following holds:

1. The impact of a change in expected TFP μ_i on the moments of log GDP is given by

$$\frac{\partial \mathbb{E}[y]}{\partial \mu_i} = \omega_i \quad \frac{\partial \mathbb{V}}{\partial \mu_i} = 0$$

2. The impact of a change in volatility Σ_{ij} on the moments of log GDP is given by

$$\frac{\partial \mathbb{E}[y]}{\partial \Sigma_{ij}} = 0 \quad \frac{\partial \mathbb{V}}{\partial \Sigma_{ij}} = \omega_i \omega_j$$

4 Firm decisions

Log of those things Log of the stochastic discount factor

$$\lambda(\alpha^*) = \log \Lambda(\alpha^*)$$

The log of the unit cost

$$k_i(\alpha_i, \alpha^*) = \log K_i(\alpha_i, P^*(\alpha^*))$$

where α^* denotes the equilibrium network.

Probelm of the firm Using this notation, we can reorganize the problem of the firm (9) as

$$\alpha_i^* \in \arg\min_{\alpha_i \in \mathcal{A}_i} \mathbb{E}[k_i(\alpha_i, \alpha^*)] + \mathrm{Cov}[\lambda(\alpha^*), k_i(\alpha_i, \alpha^*)] \tag{15}$$

Combining the equation with (5) we can write $\lambda = \log(\Lambda)$ as

$$\lambda(\alpha^*) = -(1-\rho)\sum_{i=1}^{n} \beta_i p_i(\alpha^*)$$

Taking the log of (8) yields

$$k_i(\alpha_i, \alpha^*) = -(\varepsilon_i + a(\alpha_i)) + \sum_{j=1}^{n} \alpha_{ij} p_j(\alpha^*)$$

Both $\lambda(\alpha^*)$ and $k_i(\alpha_i, \alpha^*)$ are normally distributed since they are linear combinations of ε and the log price vector, which is normally distributed by Lemma 1.

Turning to the firm problem 9, we can write

$$\alpha_i^* \in \arg\min_{\alpha_i \in \mathcal{A}_i} \mathbb{E}\left[\Lambda \frac{\beta^T \mathcal{L}(\alpha^*) \mathbb{1}_i}{P_i} K_i(\alpha_i, P)\right],$$

where we have used (A.7) from Supplemental Appendix A in Kopytov et al.(2024). We can drop $\beta^T \mathcal{L}(\alpha^*) \mathbb{1}_i > 0$ since it is a deterministic scalar that does not depend on α_i. Rewriting thisequation in terms of log quantities yields

$$\alpha_i^* \in \arg\min_{\alpha_i \in \mathcal{A}_i} \mathbb{E}[k_i(\alpha_i, \alpha^*)] + \mathrm{Cov}[\lambda(\alpha^*), k_i(\alpha_i, \alpha^*)]$$

The objective function in (15) captures how beliefs and uncertainty affect the production network. Its first term implies that the firm prefers to adopt techniques that provide, in expectation, a lower unit cost of production. Taking the expected value of the log of (8), we can write this term as

$$\mathbb{E}[k_i(\alpha_i, \alpha^*)] = -\mu_i - a_i(\alpha_i) + \sum_{j=1}^{n} \alpha_{ij} \mathbb{E}[p_j]$$

Thus we could substitute $k_i(\alpha_i, \alpha^*)$ to the (15):

$$\mathbb{E}[k_i(\alpha_i, \alpha^*)] = \mathbb{E}[\log K_i(\alpha_i, \alpha^*)] = \mathbb{E}[-\varepsilon_i - \log A_i(\alpha_i) + \sum_{j=1}^{n} \alpha_{ij} P_j]$$

$$= -\mu_i - a_i(\alpha_i) + \sum_{j=1}^{n} \alpha_{ij} \mathbb{E}[P_j]$$

By the definition $a(\alpha) = (\log A_i(\alpha_i), \cdots, \log A_n(\alpha_n))$

so that, unsurprisingly, the firm prefers techniques that have high productivity ai and that rely on inputs that are expected to be cheap.

The second term in (15) captures the importance of aggregate risk for the firm's decision. It implies that the firm prefers to have a low unit cost in states of the world in which the marginal utility of consumption is high. As a result, the coefficient of risk aversion ρ of the household indirectly determines how risk-averse firms are. We can expand this term as

$$\mathrm{Cov}[\lambda, k_i] = \mathrm{Corr}[\lambda, k_i] \sqrt{\mathbb{V}[\lambda]} \sqrt{\mathbb{V}[k_i]}$$

which implies that the firm tries to minimize the correlation of its unit cost with λ. Furthermore, since prices and GDP tend to move in opposite directions (see Lemma 1), $\mathrm{Corr}[\lambda, k_i]$ is typically positive, and so firms seek to minimize the variance of their unit cost. This has several implications for their choice of suppliers. To see this, we can use (8) to write

$$\mathbb{V}[k_i(\alpha_i, \alpha)] = \mathbb{V}[\log K_i(\alpha_i, \alpha)] = \mathbb{V}[-\varepsilon_i - \log A_i(\alpha_i) + \sum_{j=1}^{n} \alpha_{ij} P_j]$$

$$= \Sigma_{ii} + \sum_{j=1}^{n} \alpha_{ij} \mathbb{V}[p_j] + \sum_{j \neq k} \alpha_{ij} \alpha_{ik} \mathrm{Cov}[p_j, p_k] + 2\mathrm{Cov}\left[-\varepsilon_i, \sum_{j=1}^{n} \alpha_{ij} p_j\right]$$

Thus we could conclude:

$$\mathbb{V}[k_i(\alpha_i, \alpha)] = \sum_{j=1}^{n} \alpha_{ij} \mathbb{V}[p_j] + \sum_{j \neq k} \alpha_{ij} \alpha_{ik} \mathrm{Cov}[p_j, p_k] + 2\mathrm{Cov}\left[-\varepsilon_i, \sum_{j=1}^{n} \alpha_{ij} p_j\right] + \Sigma_{ii} \tag{16}$$

Lemma 2. In equilibirum, the technique choice problem of the representative firm in sector i is

$$\alpha_i^* \in \underset{\alpha_i \in \mathcal{A}_i}{\arg\max} a_i(\alpha_i) - \sum_{j=1}^{n} \alpha_{ij} \mathcal{R}_j(\alpha^*) \tag{17}$$

where

$$\mathcal{R}(\alpha^*) = \mathbb{E}[p(\alpha^*)] + \operatorname{Cov}[p(\alpha^*), \lambda(\alpha^*)] \tag{18}$$

is the vector of equilibirum risk-adjusted price, and where

$$\mathbb{E}[p(\alpha^*)] = -\mathcal{L}(\alpha^*)(\mu + a(\alpha^*)) \quad \operatorname{Cov}[p(\alpha^*), \lambda(\alpha^*)] = (\rho - 1)\mathcal{L}(\alpha^*)\Sigma[\mathcal{L}(\alpha^*)]^T \beta$$

First-order Condition Se can take the first-order condition for an interior solution of problem (17) and use the implicit function theorem to write

$$\frac{\partial \alpha_{ij}}{\partial \mathcal{R}_k} = [H_i^{-1}(\alpha_i)]_{jk} \tag{19}$$

where H_i^{-1} is the inverse of the Hessian matrix of a_i and where $[\cdot]_{jk}$ denotes its element j, k. This equation implies that if a good k becomes marginally more expensive or more risky (higher $\mathcal{R}_k$), firm i responds by changing its share α_{ik} of good k by $[H_i^{-1}(\alpha_i)]_{kk}$. Since a_i is strictly concave by Assumption 1, the diagonal elements of H_i^{-1} are negative, and so a higher $\mathcal{R}_k$ always leads to a decline in α_{ik}. The size of that decline depends on the curvature of a_i.

Substitutes and Complements Whether the increase in $\mathcal{R}_k$ leads to a decline or an increase in the share of other inputs $j \neq k$ depends on whether the shares of j and k are complements or substitutes in the production of good i. If $[H_i^{-1}]_{jk} > 0$ we say that they are **substitutes**, and in that case a higher risk-adjusted price $\mathcal{R}_k$ leads to an increase in α_{ij}. As the firm decreases α_{ik}, the incentives embedded in a_i to increase α_{ij} get stronger, and the firm substitutes α_{ij} for α_{ik}. In contrast, if $[H_i^{-1}]_{jk} < 0$ we say that the shares of j and k are **complements**, and an increase in $\mathcal{R}_k$ leads to a decline in α_{ij}. One sufficient condition for a Hessian matrix H_i to feature complementarities for all sectors is $[H_i]_{jk} \geqslant 0$ for all $j \neq k$.

Example: Substitutability and complementarity in partial equilibrium

To show how the substitution patterns embedded in ai affect technique choices, we can revisit the car manufacturer example from the introduction. Suppose that this manufacturer primarily uses steel (input

1) to produce cars, and that it relies on equipment (input 2) such as milling machines and lathes to transform raw steel into usable components. As before, the manufacturer also has the option to purchase carbon fiber (input 3) to replace steel components if needed. It would be natural to endow this manufacturer (sector $i = 4$) with a TFP shifter function of the form

$$a_4(\alpha_4) = -\sum_{j=1}^{4} \kappa_j(\alpha_{4j} - \alpha_{4j}^{\circ})^2 - \psi_1(\alpha_{41} - \alpha_{42})^2 - \psi_2[(\alpha_{41} + \alpha_{43}) - (\alpha_{41}^{\circ} + \alpha_{43}^{\circ})]^2, \tag{20}$$

where $\kappa_j > 0$, $\psi_1 > 0$ and $\psi_2 > 0$. From the second term, we see that any increase in the share α_{41} of steel would incentivize the firm to increase the share α_{42} of steel machinery as well. Inputs 1 and 2 are therefore complements in the production of cars. In contrast, the third term implies that any increase in the share α_{41} of steel would make it optimal to reduce the share α_{43} of carbon fiber, and so the shares of inputs 1 and 3 are substitutes. These patterns can be confirmed by computing the inverse Hessian of a_4 directly and inspecting the off-diagonal terms. The parameters $\psi_1 > 0$ and $\psi_2 > 0$ determine the strength of these substitution-complementarity patterns.

Figure 1 shows what happens to the production technique chosen by this car manufacturer if the risk-adjusted price of steel increases. In panel (a) the increase in $\mathcal{R}_1$ comes from a higher expectedprice $\mathbb{R}[p_1]$, while in panel (b) the price of steel becomes more volatile (higher $\mathbb{V}[p_1]$). Naturally, when the risk-adjusted price of steel rises, the manufacturer relies less on steel in production, and α_{41} falls. Since steel machinery is only useful when steel is used in production, the share α_{42} falls as well. If the increase in $\mathcal{R}_1$ is large enough, the manufacturer severs the link with its steel and steelmachinery suppliers completely so that both $\alpha_{41} = \alpha_{42} = 0$. At the same time, as steel becomes more expensive in ris-adjusted terms, the firm finds a carbon fiber supplier and progressively increases the share α_{i3}.

5 Equilibrium existence, uniqueness and efficiency

5.1 The efficient allocation

Lemma 3. An efficient production network α^* solves

$$\mathcal{W} := \max_{\alpha \in \mathcal{A}} W(a, \mu, \Sigma)$$

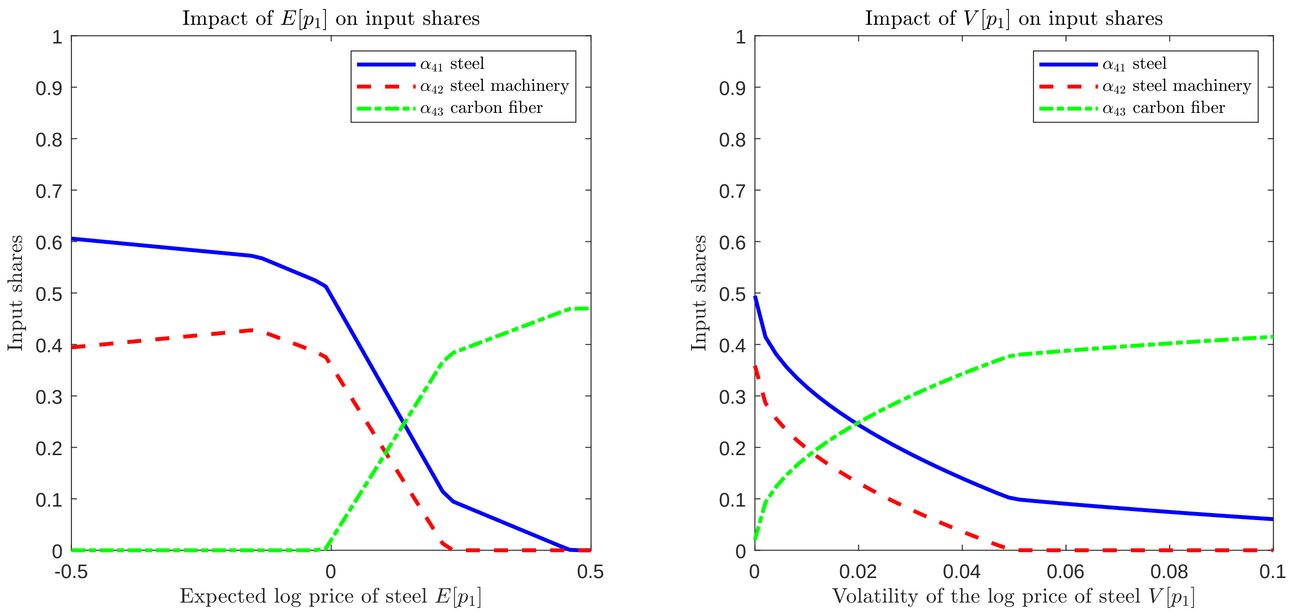
where $\mathcal{W}$ is a measure of the welfare of the household, and where

$$W(a, \mu, \Sigma) := \mathbb{E}[y(\alpha)] - \frac{1}{2}(\rho - 1)\mathbb{V}[y(\alpha)] \tag{21}$$

Risk aversion parameter

is a welfare under a given network α.

Figure 1: Impact of rising the risk-adjusted price of steel



Recasting household welfare in terms of Domar weights

Since Domar weights play a crucial role in determining the expected value and the variance of GDP, it will be useful to recast the problem of the social planner in the space of ω. Using (14), we can write the objective function (21) as

$$W(a, \mu, \Sigma) := \mathbb{E}[y(\alpha)] - \frac{1}{2}(\rho - 1)\mathbb{V}[y(\alpha)] = \omega(\alpha)^T(\mu + a(\alpha)) - \frac{1}{2}(\rho - 1)\omega(\alpha)^T\Sigma\omega(\alpha)$$

Thus we conclude that:

$$\omega^T\mu + \omega^T a(\alpha) - \frac{1}{2}(\rho - 1)\omega^T\Sigma\omega \tag{22}$$

The only term in this expression that does not depend exclusively on ω is $\omega^T a(\alpha)$, which corresponds to the contribution of the TFP shifter functions $(a_1, \cdots, a_n)$ to aggregate TFP. We want to write this object in terms of ω alone. For that purpose, notice that several networks α areconsistent with a given Domar weight vector ω, but that not all of them are equivalent in termsof welfare. Indeed, to achieve a given ω the planner will only select the network α that maximizes welfare, which amounts to maximizing $\omega^T a(\alpha)$.

Formally, consider the optimization problem

$$\bar{a}(\omega) := \max_{\alpha \in \mathcal{A}} w^T a(\alpha) \tag{23}$$

subject to the definition of the Domar weights given by $\omega^T = \beta^T \mathcal{L}(\alpha)$. We refer to the value function $\bar{a}$ as the aggregate TFP shifter function. It provides the maximum value of TFP $\omega^T a(\alpha)$ that can be achieved under the constraint that the Domar weights must be equal to some given vector ω. We denote by $\alpha(\omega)$ the solution to (23). Since both $\bar{a}(\omega)$ and $\alpha(\omega)$ depend exclusively on the TFP shifter functions $(a_1, \cdots, a_n)$ and on the preference vector β, these two functions will be invariant, for a given ω, to the changes in beliefs (μ, Σ) that we consider in the next sections.

Example.

We can solve explicitly for $\bar{a}(\omega)$ and $\alpha(\omega)$ under the quadratic TFP shifter function specified in (2). At an interior solution $\alpha \in \text{int}\mathcal{A}$, the optimal production network $\alpha(\omega)$ that solves (23) for a given vector of Domar weights ω is

$$\alpha_i(\omega) - \alpha_i^\circ = H_i^{-1} \left(\sum_{j=1}^{n} \omega_j H_j^{-1} \right)^{-1} \left(\omega - \beta - \sum_{j=1}^{n} \omega_j \alpha_j^\circ \right), \tag{24}$$

for all i, and the associated value function $\bar{a}$ is

$$\bar{a}(\omega) = \frac{1}{2} \sum_{i=1}^{n} \omega_i (\alpha_i(\omega) - \alpha_i^\circ)^T H_i (\alpha_i(\omega) - \alpha_i^\circ). \tag{25}$$

Corollary 2. The efficient Domar Weight vector ω^* solves

$$\mathcal{W} = \max_{w \in \mathcal{O}} \underbrace{\omega^T \mu + \bar{a}(\omega)}_{\mathbb{E}[y]} - \frac{1}{2}(\rho - 1) \underbrace{\omega^T \Sigma \omega}_{\mathbb{V}[y]} \tag{26}$$

where $\mathcal{O} = \{\omega \in \mathbb{R}_+^n : \ \omega \geqslant \beta \text{ and } 1 \geqslant \omega^T(\mathbb{1} - \bar{\alpha})\}$ and $\bar{a}(\omega)$ is given by (23)

Lemma 4. The objective function of the planner's problem (26) is strictly concave. Furthermore, there is a unique vector of Domar weights ω^* that solves that problme, and there is a unique production network $\alpha(\omega^*)$ associated with that solution.

5.2 Fundamental properties of the equilibrium

Proposition 1. There exists a unique equilibrium, and it is efficient.

6 Beliefs and the production network

In this section, we characterize how beliefs (μ, Σ) affect the equilibrium production network. We begin with a general result that describes how a change in a sector's risk or expected TFP impacts its own Domar weight. We then provide an expression that characterizes how the full vector ofDomar weights responds to a marginal change in (μ, Σ). Finally, we investigate how beliefs affect the structure of the underlying production network α. As we only consider the equilibrium networkfrom now on, we lighten the notation by dropping the superscript $*$ when referring to equilibrium variables.

6.1 Domar weights

In contrast, when the network isendogenous, they are equilibrium objects that vary with (μ, Σ). The next proposition describes the relationship between these quantities.

Proposition 2. The Domar weight ω_i of sector i is (weakly) increasing in μ_i and (weakly) decreasing in Σ_{ii}.

Risk-adjusted productivity shocks

Proposition 2 describes how the Domar weight of a sector responds to a change in its own TFP process, and it holds generally. At an interior equilibrium, we can also characterize how any change in beliefs affects the full vector ω. For that purpose, we introduce a risk-adjusted version of the productivity vector ε defined as

$$\mathcal{E} = \underbrace{\mu}_{\mathbb{E}[\varepsilon]} - \underbrace{(\rho - 1)\Sigma\omega}_{\mathrm{Cov}[\varepsilon,\lambda]} \tag{27}$$

The vector $\mathcal{E}$ captures how higher exposure to the productivity process ε_i affects the representative household's utility. It depends on how productive each sector i is in expectation, and on how its ε_i covaries with the stochastic discount factor λ. If we denote by $\mathbb{1}_i$ the column vector with a 1 as ith element and zeros elsewhere, we can write

$$\frac{\partial \mathcal{E}}{\partial \mu_i} = \mathbb{1}_i, \tag{28}$$

such that an increase in μ_i makes sector i more attractive. It however leaves the risk-adjusted TFP of other sectors unchanged. Similarly, for a change in Σ_{ij}, we can compute

$$\frac{\partial \mathcal{E}}{\partial \Sigma_{ij}} = -\frac{1}{2}(\rho - 1)(\omega_j \mathbb{1}_i + \omega_i \mathbb{1}_j) \tag{29}$$

Using the definition of $\mathcal{E}$, we can write the first-order condition of the planner's problem (26) at an interior solution as

$$\nabla \bar{a}(\omega) + \mathcal{E} = 0 \tag{30}$$

where ∇ is the gradient of the aggregate TFP shifter function $\bar{a}$. This first-order condition shows that the planner balances the benefit of a sector in terms of risk-adjusted TFP against its impact on the aggregate TFP shifter.

Proposition 3. Let γ denote either the mean μ_i or an element of the covariance matrix Σ_{ij}. If $\omega \in \text{int}\mathcal{O}$, then the response of the equilibrium Domar weights to a change in γ is given by

$$\frac{d\omega}{d\gamma} = \underbrace{-\mathcal{H}^{-1}}_{\text{propagation}} \times \underbrace{\frac{\partial \mathcal{E}}{\partial \gamma}}_{\text{impulse}} \tag{31}$$

where the $n \times n$ negative definite matrix $\mathcal{H}$ is given by

$$\mathcal{H} = \nabla^2 \bar{a} + \frac{\partial \mathcal{E}}{\partial \omega} \tag{32}$$

and where the matrix $\nabla^2 \bar{a}$ is the Hessian of the aggregate TFP shifter function $\bar{a}$, and $\frac{\partial \mathcal{E}}{\partial \omega} = -\frac{d\text{Cov}[\varepsilon,\lambda]}{d\omega} = -(\rho-1)\Sigma$ is the Jacobian matrix of the risk-adjusted TFP vector $\mathcal{E}$.

The response of the Domar weights to a change in beliefs, as given by (31), can be decomposed into an impulse component and a propagation component. The impulse captures the direct impact of the change on risk-adjusted TFP. It is simply given by the partial derivative of $\mathcal{E}$ with respect to the moment of interest (see (28) and (29) above). This impulse is then propagated through $\mathcal{H}^{-1}$ to capture its full equilibrium effect on the Domar weights.

Global complements and substitutes Just as $\mathcal{H}_i^{-1}$ captured local substitution patterns between inputs in the problem of firm i, $\mathcal{H}^{-1}$ captures global, economy-wide substitution patterns between sectors.

If $\mathcal{H}_{ij}^{-1} < 0$, we say that i and j are **global complements**. If instead $\mathcal{H}_{ij}^{-1} > 0$, we say that i and j are **global substitutes**.

The following corollary justifies this terminology by showing that the sign of $\mathcal{H}_{ij}^{-1}$ is sufficient to characterize how Domar weights respond to a change in the productivity process.

Corollary 3. If $w \in \text{int}\mathcal{O}$, then the following holds.

1. An increase in the expected value μ_i or a decline in the variance Σ_{ii} leads to an increase in ω_j if i and j are global complements, and to a decline in ω_j if i and j are global substitutes.

2. An increase in the covariance Σ_{ij}, $i \neq j$, leads to a decline in ω_k if k is global complement with i and j, and to an increase in ω_k if k is global substitute with i and j.

Σ and global substition patterns The following lemma describes how an increase in covariance Σ_{ij} between any two sectors affects the degree of global substitution between them.

Lemma 5. An increase in the covariance Σ_{ij} induces stronger global substitution between i and j, in the sense that $\frac{\partial \mathcal{H}_{ij}^{-1}}{\partial \Sigma_{ij}} > 0$.

Intuitively, if the correlation between ε_i and ε_j becomes larger, the planner has stronger incentives to lower ω_j after an increase in ω_i in order to reduce aggregate risk. From (32), we see that the strength of that diversification mechanism depends on the household's risk aversion through ρ.

$\nabla^2 \bar{a}$ and global substitution patterns The next lemma establishes sufficient conditions under which local complementarities translate into global complementarities.

Lemma 6. Suppose that all input shares are (weak) local complements in the production of all goods, that is $[H_i^{-1}]_{kl} \leqslant 0$ for all i and all $k \neq l$. If $\alpha \in \text{int}\mathcal{A}$, there exists a scalar $\bar{\Sigma} > 0$ such that if $\|\Sigma\| \leqslant \bar{\Sigma}$, all sectors are global complements, that is $\mathcal{H}_{ij}^{-1} < 0$ for all $i \neq j$.

Impact of Lemma 6

1. **Generation of Global Complementarities**: Even if the local TFP shifter functions are neutral (i.e., $\left[H_i^{-1}\right]_{kl} = 0$ for all i and $k \neq l$), the equilibrium forces of the model generate global complementarities between sectors. This means that the model itself induces sectors to be globally complementary without requiring local TFP shifter functions to exhibit local complementarities.

2. **Equilibrium Forces** Suppose a sector i becomes more attractive, for instance due to an increase in μ_i. Any other sector j that relies on i (either directly or indirectly, if $L_{ji} > 0$) would benefit from that change and also become more attractive. This triggers an increase in Domar weights throughout the network and a shift away from labor, generating global complementarities between sectors.

3. **Policy and Practical Applications** Understanding the conditions under which local complementarities translate into global complementarities can help in formulating more effective economic policies, especially regarding resource allocation and inter-sector coordination. This is crucial for improving overall economic efficiency and welfare.

4. **Role of Covariance Matrix (Σ)** The lemma highlights that the degree of global substitution or complementarity between sectors can be influenced by the covariance matrix Σ. If Σ is sufficiently small, local complementarities can lead to global complementarities, while larger Σ might induce stronger global substitution forces due to diversification effects.

Parametrize H_i Let

$$H_i^{-1} = \begin{bmatrix} -1 & \frac{s}{n-1} & \cdots & \frac{s}{n-1} \\ \frac{s}{n-1} & -1 & & \vdots \\ \vdots & & \ddots & \frac{s}{n-1} \\ \frac{s}{n-1} & \cdots & \frac{s}{n-1} & -1 \end{bmatrix} \tag{33}$$

where we impose $-(n-1) < s < 1$ to guarantee that H_i^{-1} is negative definite. When $s < 0$ all input shares are complements in the production of good i, and when $s > 0$ they are substitutes. The next lemma describes sufficient conditions under which local substitution imply global substitution.

Lemma 7. Suppose that all the TFP shifter functions $(a_1, \cdots, a_n)$ take the form (2), with $\alpha_i^\circ = \alpha_j^\circ$ for all i, j, and that H_i^{-1} is of the form (33) for all i. If $\alpha \in \text{int}\mathcal{A}$, there exists a scalar $\bar{\Sigma} > 0$ and a threshold $0 < \bar{s} < 1$ such that if $\|\Sigma\| \leqslant \bar{\Sigma}$ and $s > \bar{s}$, then all sectors are global substitutes, that is $\mathcal{H}_{ij}^{-1} > 0$ for all $i \neq j$.

An approximate equation for the equilibrium Domar weights This section discusses how to derive an approximate equation for the equilibrium Domar weights using a Taylor expansion of $\nabla\bar{a}$. The key steps and impacts are outlined as follows:

First, we define the ideal shares α°, which maximize the values of the TFP shifters $(a_1, \dots, a_n)$. Based on this, we can write:

$$\nabla \bar{a}(\omega) \approx \nabla \bar{a}(\omega^\circ) + \nabla^2 \bar{a}(\omega^\circ)(\omega - \omega^\circ) \tag{34}$$

This approximation is accurate if the cost of deviating from the ideal shares embedded in the local TFP shifters is large.

Using this approximation, the first-order condition (30) becomes linear in ω, allowing us to solve for the equilibrium Domar weights.

Lemma 8. If $\omega \in \text{int}\mathcal{O}$, the equilibrium Domar weights are approximately given by:

$$\omega = \omega^\circ - [\mathcal{H}^\circ]^{-1}\mathcal{E}^\circ + O\left(\|\omega - \omega^\circ\|^2\right) \tag{35}$$

where the superscript $\circ$ indicates that $\mathcal{H}$ and $\mathcal{E}$ are evaluated at ω°.

Impacts of Lemma 8

1. **Global Substitution Patterns** This approximation shows that the equilibrium Domar weights can be explained in terms of the global substitution patterns embedded in $[\mathcal{H}^\circ]^{-1}$ and the expected attractiveness of all sectors, captured by the risk-adjusted productivity $\mathcal{E}^\circ$.

2. **Inter-Sector Interactions** If a sector i is endowed with a productivity process that is high in expectation or has a low covariance with the stochastic discount factor, $\mathcal{E}_i^\circ$ will be large. Since the diagonal elements of $[\mathcal{H}^\circ]^{-1}$ are negative, ω_i tends to be larger than ω_i°.

3. **Relative Weight Changes** A large $\mathcal{E}_i^\circ$ also contributes to increasing the Domar weights of all sectors that are global complements with i and to decreasing the Domar weights of sectors that are global substitutes with i.

6.2 The production network

Proposition 4. If $\alpha \in \text{int}\mathcal{A}$, there exists a scalar $\bar{\Sigma} > 0$ such that if $\|\Sigma\| \leqslant \bar{\Sigma}$ the following holds.

1. (Complementarity) Suppose that input shares are local complements in the production of good i, that is $[H_i^{-1}]_{kl} < 0$ for all $k \neq l$. Then a beneficial change to k $(\partial \mathcal{E}_k / \partial \gamma > 0)$ increases α_{ij} for all j.

2. (Substitution) Suppose that the conditions of Lemma 7 about the TFP shifters $(a_1, \cdots, a_n)$ hold. Then there exists a threshold $0 < \bar{s} < 1$ such that if $s > \bar{s}$, a beneficial change to k $(\partial \mathcal{E}_k / \partial \gamma > 0)$ decreases α_{ij} for all i and all $j \neq k$, and increases α_{ik} for all i.

Proposition 4 illustrates the impact of complementarity and substitution of input shares on the adjustment of production networks. When input shares are locally complementary in the production of a product, a beneficial change to one input increases its share in the production of all products. Conversely, in the presence of strong substitution effects, a beneficial change to one input decreases the shares of other inputs in production while increasing its own share.

An approximate equation for the equilibrium production network

As for the Domar weights, one must in general use numerical methods to find the equilibrium network α. We can, however, derive an approximation for the equilibrium production network when the cost of deviating from the ideal shares α° is large. Specifically, let $a_i(\alpha_i) = \bar{\kappa} \times \hat{a}_i(\alpha_i)$, where $\hat{\alpha}$ does not depend on κ, and suppose that $\alpha_i^\circ \in \text{int}\mathcal{A}_i$. The parameter $\hat{\kappa} > 0$ captures how costly it is for the firms to deviate from α° in terms of TFP loss. When $\hat{\kappa}$ is large, we can use perturbation theory to derive an approximate equation for α.

Lemma 9. If $\alpha \in \text{int}\mathcal{A}$, the equilibrium input shares in sector i are approximately given by

$$\alpha_i = \alpha_i^\circ + \bar{\kappa}^{-1} \left(\hat{H}_i^\circ\right)^{-1} \mathcal{R}^\circ + O(\kappa^{-2}) \tag{36}$$

where $\hat{H}_i^\circ$ is the Hessian of $\hat{a}_i$ at α_i°, and where the vector of risk-adjusted prices at α° is given by

$$\mathcal{R}^\circ = -\mathcal{L}\mu + (\rho - 1)\mathcal{L}^\circ \Sigma \omega^\circ$$

Lemma 9 primarily addresses the approximate solution for the production network when the cost function is nonlinear. Specifically, when it is costly for firms to deviate from the ideal shares α°, the equilibrium production network can be approximated using perturbation theory. Equation (36) provides an approximation indicating that the equilibrium input shares α_i depend on the risk-adjusted prices $\mathcal{R}^\circ$. This result demonstrates that when the cost of deviating from the ideal shares is high, the equilibrium production network can be approximated by evaluating the equilibrium prices as if firms chose the ideal shares.

Example: cascading link destruction

The example of "cascading link destruction" discusses how an increase in uncertainty in a single sector can trigger a chain reaction throughout the production network. Specifically, when the volatility of a sector increases, multiple producers sequentially switch to more stable suppliers, causing a series of adjustments in the production network.

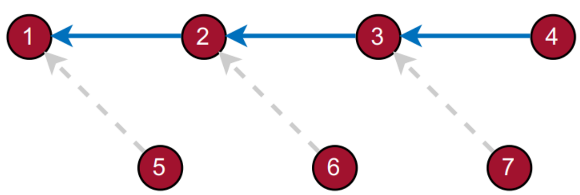
The specific example is as follows:

1. In a low-uncertainty state (left figure): Firms in sectors 1 to 3 directly or indirectly rely on sector 4 as a supplier.

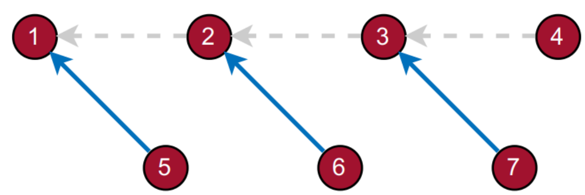
2. In a high-uncertainty state (right figure): As the uncertainty in sector 4 increases, firms in sector 3, seeking a more stable supply, switch to using inputs from sector 7. This change implies that firms in sector 2, to avoid risk, switch to using inputs from sector 6, and so on, creating a cascade of adjustments.

Through this example, the paper demonstrates how the production network adjusts in response to changes in uncertainty. These adjustments not only affect the directly related firms but also propagate through the supply chain, impacting firms far removed from the initial shock.

Figure 2: Cascading impact of a change in Σ_{44}



(a) Low uncertainty about ε_4



(b) High uncertainty about ε_4

We can interpret this cascading network adjustment through the lens of Lemma 9. Differentiating the expression with respect to Σ_{44} yields

$$\frac{d\alpha_{ij}}{d\Sigma_{44}} = \bar{\kappa}^{-1}(\rho-1)\omega_4^\circ \left(\underbrace{\left[\left(\hat{H}_i^\circ\right)^{-1}\right]_{jj} \mathcal{L}_{j4}^\circ}_{\text{direct effect of } \Sigma_{44} \text{ on } j} + \underbrace{\sum_{l\neq j} \left[\left(\hat{H}_i^\circ\right)^{-1}\right]_{jl} \mathcal{L}_{jl}^\circ}_{\text{indirect effect of } \Sigma_{44} \text{ through other suppliers } l\neq j} \right) + O(\bar{\kappa}^{-2}) \tag{37}$$

Equation (37) states that if a firm j relies on sector 4 as an input (either immediate or distant, such that $\mathcal{L}_{j4}^\circ > 0$), an increase in Σ_{44} makes j less attractive. This direct effect pushes α_{ij} down (recall that $[H_i^\circ]_{jj} < 0$ by the concavity of a_i). There is also an indirect effect that operates through the second term in (37). If another sector $l \neq j$ also relies on 4 ($\mathcal{L}_{l4}^\circ > 0$), then an increase in Σ_{44} makes l less attractive as well. This indirect channel can lead to either a decrease or an increase in α_{ij}, depending on whether j and l are complements or substitutes in the production of i; that is, whether $[(H_i^\circ)^{-1}]_{jl}$ is negative or positive.

7 Implications for GDP and welfare

Proposition 5. Let γ denote either the mean μ_i or an element of the covariance matrix Σ_{ij}. Under an endogenous network, welfare responds to a marginal change in γ as if the network were fixed at its equilibrium value α^*, that is

$$\frac{d\mathcal{W}(\mu,\Sigma)}{d\gamma} = \frac{\partial W(\alpha^*,\mu,\Sigma)}{\partial\gamma}$$

Let α^* be the equilibrium network, i.e., $\alpha^* = \alpha(\mu,\Sigma)$. When we make a small change to γ, the equilibrium network will adjust to accommodate the new γ. However, Proposition 5 states that the effect of this adjustment on the marginal change can be neglected.

While this proposition shows that the flexibility of the network plays no role for the response of welfare to a marginal change in beliefs, this is generally not true for non-infinitesimal changes. In that case, shifts in (μ,Σ) that are beneficial to welfare are amplified, compared to the fixed-network benchmark, while changes that are harmful are dampened (see Proposition 2). Indeed, if we denote by $\alpha^*(\mu,\Sigma)$ the equilibrium production network under (μ,Σ) and by $W(\alpha,\mu,\Sigma)$ welfare under a network α, we can write that the difference in welfare after a change in beliefs from (μ,Σ) to (μ',Σ') satisfies the inequality

$$\underbrace{\mathcal{W}(\mu',\Sigma') - \mathcal{W}(\mu,\Sigma)}_{\text{Change in welfare under a flexible network}} \geqslant \underbrace{W(\alpha^*(\mu,\Sigma),\mu',\Sigma') - W(\alpha^*(\mu,\Sigma),\mu,\Sigma)}_{\text{Change in welfare under a fixed network}}. \tag{38}$$

Corollary 4. The impact of an increase in μ_i on welfare is given by

$$\frac{d\mathcal{W}}{d\mu_i} = \omega_i \tag{39}$$

and the impact of an increase in Σ_{ij} on welfare is given by

$$\frac{d\mathcal{W}}{d\Sigma_{ij}} = -\frac{1}{2}(\rho - 1)\omega_i\omega_j \tag{40}$$

7.1 Beliefs and GDP

Proposition 6. The presence of uncertainty lowers expected log GDP, in the sense that $\mathbb{E}[y]$ is largest when $\Sigma = 0$.

This proposition follows directly from Lemma 3. Without uncertainty ($\Sigma = 0$), the variance $\mathbb{V}[y]$ of log GDP is zero for all networks $\alpha \in \mathcal{A}$. The social planner then maximizes $\mathbb{E}[y]$ only. When, instead, the productivity vector ε is uncertain ($\Sigma \neq 0$), the planner also seeks to lower $\mathbb{V}[y]$ which necessarily lowers expected log GDP in equilibrium.

Corollary 5. Let γ denote either the mean μ_i or an element of the covariance matrix Σ_{ij}. The equilibrium response to a change in beliefs γ must satisfy

$$\underbrace{\frac{d\mathbb{E}[y]}{d\gamma} - \frac{\partial\mathbb{E}[y]}{\partial\gamma}}_{\text{Excess response of } \mathbb{E}[y]} = \frac{1}{2}(\rho - 1)\underbrace{\left(\frac{d\mathbb{V}[y]}{d\gamma} - \frac{\partial\mathbb{V}[y]}{\partial\gamma}\right)}_{\text{Excess response of } \mathbb{V}[y]} \tag{41}$$

Corollary 5 is a direct consequence of Proposition 5. Since the response of welfare to a marginal change in beliefs must be the same under a flexible and a fixed network, a larger increase in $\mathbb{E}[y]$ under a flexible network must come at the cost of alarger increase in the variance $\mathbb{V}[y]$.

Proposition 7. If $\omega \in \text{int}\mathcal{O}$, the following holds.

1. The impact of an increase in μ_i on log GDP is given by

$$\frac{d\mathbb{E}[y]}{d\mu_i} = \underbrace{\omega_i}_{\text{Fixed network}} - (\rho - 1)\omega^T \Sigma \mathcal{H}^{-1}\frac{\partial \mathcal{E}}{\partial \mu_i}, \quad \text{and} \quad \frac{d\mathbb{V}[y]}{d\mu_i} = \underbrace{0}_{\text{Fixed network}} - 2\omega^T \Sigma \mathcal{H}^{-1}\frac{\partial \mathcal{E}}{\partial \mu_i}.$$

2. The impact of an increase in Σ_{ij} on log GDP is given by

$$\frac{d\mathbb{E}[y]}{d\Sigma_{ij}} = \underbrace{0}_{\text{Fixed network}} - (\rho - 1)\omega^T \Sigma \mathcal{H}^{-1}\frac{\partial \mathcal{E}}{\partial \Sigma_{ij}}, \quad \text{and} \quad \frac{d\mathbb{V}[y]}{d\Sigma_{ij}} = \underbrace{\omega_i\omega_j}_{\text{Fixed network}} - 2\omega^T \Sigma \mathcal{H}^{-1}\frac{\partial \mathcal{E}}{\partial \Sigma_{ij}}.$$

Corollary 6. Without uncertainty ($\Sigma = 0$) the moments of GDP respond to changes in beliefs as if the network were fixed, such that

$$\frac{d\mathbb{E}[y]}{d\mu_i} = \frac{\partial \mathbb{E}[y]}{\partial \mu_i} = \omega_i, \quad \text{and} \quad \frac{d\mathbb{V}[y]}{d\Sigma_{ij}} = \frac{\partial \mathbb{V}[y]}{\partial \Sigma_{ij}} = \omega_i \omega_j$$

Corollary 7. Suppose that $\omega \in \text{int}\mathcal{O}$. There exists a threshold $\bar{\Sigma} < 0$ such that if $\Sigma_{kl} > \bar{\Sigma}$ for all k, l, then the following holds.

1. If all sectors are global complements with sector i, that is $\mathcal{H}^{-1}_{ik} < 0$ for $k \neq i$, then

$$\frac{d\mathbb{E}[y]}{d\mu_i} = \frac{\partial \mathbb{E}[y]}{\partial \mu_i} > \omega_i, \quad \text{and} \quad \frac{d\mathbb{V}[y]}{d\Sigma_{ij}} = \frac{\partial \mathbb{V}[y]}{\partial \mu_i} > 0$$

2. If all sectors are global complements with sectors i and j, that is $\mathcal{H}^{-1}_{ik} < 0$ and $\mathcal{H}^{-1}_{jk}$ for $k \neq i, j$, then

$$\frac{d\mathbb{E}[y]}{d\mu_i} = \frac{\partial \mathbb{E}[y]}{\partial \Sigma_{ij}} < 0, \quad \text{and} \quad \frac{d\mathbb{V}[y]}{d\Sigma_{ij}} = \frac{\partial \mathbb{V}[y]}{\partial \mu_i} < \omega_i \omega_j$$

Corollary 8. Suppose that $\omega \in \text{int}\mathcal{O}$. There exists a threshold $\underline{\Sigma} < 0$ and $\bar{\Sigma} > 0$ such that

1. If all sectors are global substitutes with sector i, that is $\mathcal{H}^{-1}_{ik} > 0$ for $k \neq i$, and sector i is not too risky while other sectors are sufficiently risky in the sense that $\Sigma_{ji} < \underline{\Sigma}$ for all j and $\Sigma_{jk} > \bar{\Sigma}$ for all $j, k \neq i$, then

$$\frac{d\mathbb{E}[y]}{d\mu_i} < \omega_i, \quad \text{and} \quad \frac{d\mathbb{V}[y]}{d\mu_i} < 0.$$

2. If all sectors are global substitutes with sectors i and j, that is $\mathcal{H}^{-1}_{ik} > 0$ and $\mathcal{H}^{-1}_{jk} > 0$ for $k \neq i, j$, and sectors i and j are not too risky while other sectors are sufficiently risky in the sense that $\Sigma_{li} < \underline{\Sigma}$ and $\Sigma_{lj} < \underline{\Sigma}$ for all l, and $\Sigma_{lk} > \bar{\Sigma}$ for all $l, k \neq i$ and $l, k \neq j$, then

$$\frac{d\mathbb{E}[y]}{d\Sigma_{ij}} > 0, \quad \text{and} \quad \frac{d\mathbb{V}[y]}{d\Sigma_{ij}} > \omega_i \omega_j.$$

After an increase in the TFP mean (μ_i) of a sector, the Domar weight (ω_i) of that sector increases, which pushes up the variance of log GDP ($\mathbb{V}[y]$). However, if the sector's TFP variance (Σ_{ii}) is small, the increase in $\mathbb{V}[y]$ is also small. Since other sectors are global substitutes with this sector, the increase in ω_i leads to a decline in the Domar weights of all other sectors. If the variances of these other sectors are large relative to Σii, this decline in their Domar weights results in a substantial decrease in $\mathbb{V}[y]$. According to the logic of Proposition 7, this means that the expected log GDP ($\mathbb{E}[y]$) must increase by less than the fixed-network term ω_i. Similarly, an increase in Σii leads to an increase in $\mathbb{V}[y]$ that is larger than under a fixed network. In this case, $\mathbb{E}[y]$ increases in response to the higher Σ_{ii}, indicating that uncertainty can be beneficial to expected log GDP at the margin.

Counterintuitive implications of changes in beliefs

1. **Belief Changes and GDP Response**
   (a) Corollaries 7 and 8 indicate that the response of GDP to changes in beliefs can be different from the predictions of Hulten's theorem in a fixed-network economy. The endogenous adjustment of the network can lead to more extreme outcomes.
   (b) An increase in the mean productivity (μ) of a sector can lead to a decrease in the expected log GDP ($\mathbb{E}[y]$), and an increase in the variance (Σ) of a sector can lead to a decrease in the variance of log GDP ($\mathbb{V}[y]$).
2. **Example of a Low-Productivity but Stable Producer**
   (a) Consider a producer with low but stable productivity. The high price of its goods makes it less attractive as a supplier.
   (b) If its expected productivity increases, its risk-reward profile improves, attracting more buyers. This can lead producers to move away from more productive but riskier suppliers, potentially causing expected GDP to fall.
3. **Impact of Increased Volatility** An increase in the volatility of a sector's productivity can lead to a decline in $\mathbb{V}[y]$. This is because producers may shift away from more volatile sectors, leading to a network that is less susceptible to fluctuations.

In the economy depicted in Figure 3, sectors 4 and 5 use only labor to produce, while sectors 1 to 3 can also use goods 4 and 5 as inputs. For sectors 1 to 3, goods 4 and 5 are either local substitutes (panels (a) to (c)) or local complements (panels (d) to (f)). Sector 4 is more productive and volatile than sector 5.

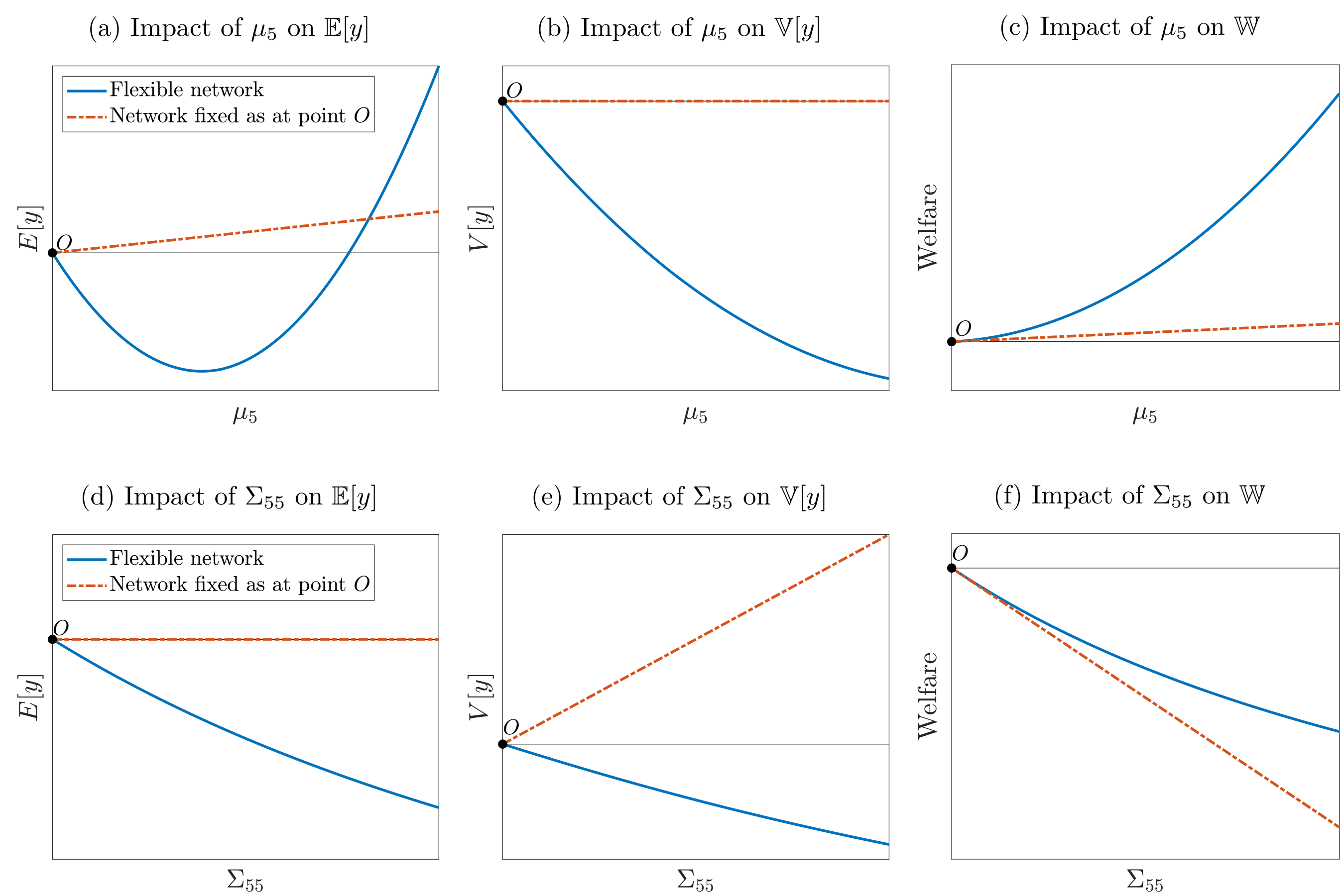
Consider the impact of a positive shock to μ_5 when inputs 4 and 5 are substitutes. Initially, the increase in μ_5 negatively impacts expected log GDP ($\mathbb{E}[y]$) because sector 5, although less productive, now offers a better risk-reward trade-off. This causes sectors 1 to 3 to shift towards good 5 and away from good 4, reducing $\mathbb{E}[y]$ since $\mu_4 > \mu_5$. V[y] also declines because sector 5 is less volatile, aligning with Proposition 7. Ultimately, the overall effect on welfare is positive, as shown in panel (c), because the welfare gain from reduced volatility outweighs the initial drop in $\mathbb{E}[y]$.

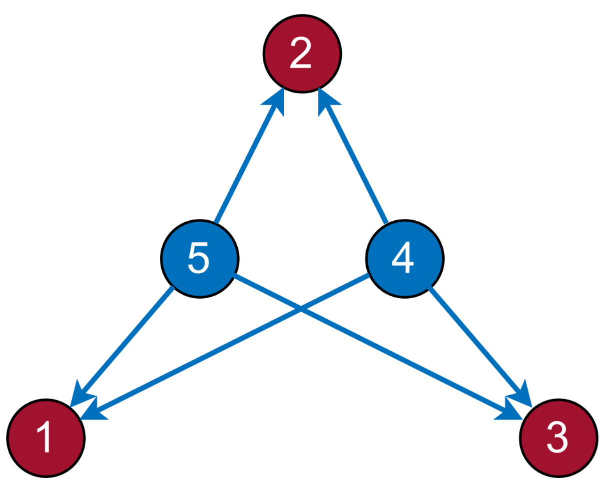
To emphasize the role of the endogenous network for this mechanism, Figure 3 also shows the effect of the same increase in μ_5 when the network is kept fixed (dashed red lines). From Corollary 1, the marginal impact of μ_5 on expected log GDP is equal to its Domar weight, and increasing μ_5 has a positive impact on $\mathbb{E}[y]$. At the same time, $\mathbb{V}[y]$ is unaffected by changes in μ. Whilean increase in μ_5 is welfare-improving in this case, the effect is less pronounced than in the flexible network economy. Indeed, in the latter case the equilibrium network adjusts precisely to maximize the beneficial impact of the change in beliefs on welfare, as implied by (38).

We can use a small variation of this economy to illustrate how an increase in an element of Σ can lower the variance of log GDP, and simultaneously lower welfare. Start again from the economy in the left column of Figure 3 (point O) but suppose that inputs 4 and 5 are complements in the production of goods 1 to 3. Consider an increase in the volatility of sector 5. In response, sectors 1 to 3 start to rely less on sector 5. But since inputs 4 and 5 are complements, sectors 1 to 3 also reduce their shares of input 4, thus increasing the overall share of labor which is a safe input. As a result, the variance of log GDP declines (panel e). Expected log GDP also goes down by Proposition 7 (panel d). The combined effect on welfare is negative, as predicted by Corollary 4 (panel f). In this case, the reorganization of the network mitigates the adverse effect of the increase in volatility on welfare. Instead, if the network is fixed, an increase in Σ_{55} does not affect expected log GDP but leads to an increase in the variance of log GDP. As a result, welfare drops substantially more than under an endogenous network, as implied by (38).

Notes: There is an arrow from j to i if $\alpha_{ij} > 0$. Household: $\rho = 2.5$ and $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3} - \varepsilon$, $\beta_4 = \beta_5 = \frac{3}{2}\varepsilon$, where $\varepsilon > 0$ is very small. $\mu = (0.1, 0.1, 0.1, 0.1, -0.08)$, Σ is diagonal, with $\mathrm{diag}(\Sigma) = (0.2, 0.2, 0.2, 0.2, 0.02)$. a is as in (2) with $\alpha^\circ_{14} = \alpha^\circ_{15} = \alpha^\circ_{24} = \alpha^\circ_{25} = \alpha^\circ_{34} = \alpha^\circ_{35} = 0.25$; all other α°_{ij} are zero. $H_4 = H_5$ are matrices with -50 on the diagonal. $H_1 = H_2 = H_3$ with $[H_1]_{11} = [H_1]_{22} = [H_1]33 = -50$, $[H_1]_{44} = [H_1]_{55} = -2$. In panels (a)-(c), μ_5 goes from -0.08 to 0.1; 4 and 5 are substitutes, $[H_1]_{45} = -1.9$. In panels (d)-(f), Σ_{55} goes from 0.02 to 0.2; 4 and 5 are complements, $[H_1]_{45} = 1.9$.

Figure 3: The non-monotone impact of beliefs on GDP





8 A basic calibration of the model

The analysis above highlights the economic forces that determine how the production network, GDP and welfare respond to changes in the productivity process. Clearly, the model is too stylized to capture all the fluctuations in the production network observed in reality, and other mechanisms, not present in our model, may also be important in practice. With that caveat in mind, we present in this section results from a basic calibration of the model to the United States economy to get a sense of the quantitative potential of our main mechanisms.

Below, we first describe how the model is parameterized and briefly go over which features of the US economy the model matches well, and in what dimensions it falls short. Finally, we explore how beliefs shape the production network and investigate how the changing structure of the network influences aggregate output and welfare in our stylized model. We keep the analysis succinct but provide more details in Appendix B.

8.1 Parametrization

The Bureau of Economic Analysis (BEA) provides U.S. sectoral input-output tables for $n = 37$ sectors at an annual frequency from 1948 to 2020. From these data, we compute the input shares α_{ijt} of each sector in each year t, the average consumption expenditure share of each sector β_i, and sectoral TFP measured as the Solow residual.

To calibrate the model, we need to make explicit assumptions about the process for TFP. For the endogenous productivity shifter $A_i(\alpha_{it})$ we adopt a particular version of form (2) which includes a diagonal component for $\bar{H}_i$ are a penalty for deviating from an ideal labor share (see (69) inthe appendix). We set the ideal shares $(\alpha_1^\circ, \cdots, \alpha_n^\circ)$ equal to the time average of the input shares observed in the data. The exogenous sectoral productivity process ε_t is assumed to follow a random walk with drift,

$$\varepsilon_t = \gamma + \varepsilon_{t-1} + u_t, \tag{42}$$

where γ is an $n \times 1$ vector of deterministic drifts and $u_t \sim \text{iid}\mathcal{N}(0, \Sigma_t)$ is a vector of shocks. We further assume that firms know γ and ε_t at time t, so that the conditional mean and the covariance of beliefs are given by $\mu_t = \gamma + \varepsilon_{t-1}$ and Σ_t. Importantly, we allow uncertainty Σ_t to vary over time and estimate it from TFP data using a rolling window that puts more weight on more recent observations.

We use a simple moment-matching strategy to pin down the 1) relative risk aversion parameter ρ of the household, 2) the TFP shifter functions $\bar{H}_i$ and 3) the time-varying beliefs (μ_t, Σ_t). We describe this procedure in Appendix B.

The calibrated coefficient of relative risk aversion $\hat{\rho}$ is 4.3, which is similar to values used orestimated in the macroeconomics literature. Our procedure also provides time-series for the vector μ_t and the matrix Σ_t, and we aggregate these variables across sectors to obtain economy-wide measures of the expected value $\bar{\mu}_t$ and the variance $\bar{\Sigma}_t$ of aggregate TFP. As we might expect, these measures are cyclical, with $\bar{\mu}_t$ falling and $\bar{\Sigma}_t$ rising during recessions. Overall, our measure of aggregate uncertainty $\bar{\Sigma}_t$ has been relatively stable since 1980, with occasional sharp spikes, most notably during the Great Recession of 2007–2009 (see Figure 5 in Appendix B.3).

We next assess how well the calibrated model fits key moments in the data. As we have seen above, the Domar weights, and how they react to changes in μ_t and Σ_t, are central for the mechanisms of the model. The model is able to roughly replicate features of the empirical Domar weights, with a cross-sectional correlation between the time-averaged Domar weights in the model and in the data of 0.96. However, the average Domar weight in the model (0.03) is lower than its data counterpart (0.05). Overall, the model can account for about 40% of the over-time standard deviation of Domar weights, which indicates that other mechanisms, such as technological progress that might expand the set of available techniques, might be at work in reality.

The mechanisms of the model predict that a decline in the expected productivity of a sector μ_i, or an increase in its variance Σ_{ii}, should push firms to reduce the importance of that sector as an input provider, leading to a decline in its Domar weight. Reassuringly, these correlations are visible in the data, where $\text{Corr}(\omega_{jt}, \mu_{jt}) = 0.1$, and $\text{Corr}(\omega_{jt}, \Sigma_{jjt}) = -0.4$. The calibrated model is also able to roughly match these correlations, and the corresponding numbers are 0.1 and -0.3.

8.2 The production network, welfare and output

To evaluate the quantitative potential of an endogenous production network for welfare and GDP, we compare the calibrated model to two sets of alternative economies. First, we compare our baseline model to an economy in which the network is kept completely fixed at its sample average. This exercise therefore informs us about the overall impact of changes in the structure of the production network. We then investigate the role of uncertainty alone in shaping the production network. We do so by considering 1) an economy in which production techniques are chosen as if $\Sigma_t = 0$, and 2) a perfect-foresight economy in which firms observe the realization of ε_t before making technique choices (the "known ε_t" economy). In both cases, uncertainty is irrelevant for decisions, and so these exercises allow us to isolate the impact of uncertainty on the production network and, through that channel, on macroeconomic aggregates.

We find that expected log GDP in the "fixed network" economy is 2.1% lower than in our baseline calibration with a flexible network. Intuitively, as some sectors become more productive over time, the goods that they produce become cheaper, and firms would like to rely more on them. With a flexible network this is possible, and the aggregate economy becomes more productive as aresult. The difference in welfare between the two models is about 2.1% as well.

When we isolate the role of uncertainty, however, these numbers become smaller. In line withthe theory, the baseline economy is on average less productive and less volatile than under the "as if $\Sigma_t = 0$" alternative but the numbers are small, on the order of 0.01% for $\mathbb{E}[y]$ and 0.10% for $\mathbb{V}[y]$. This suggests that, for most of the sample period, uncertainty is sufficiently low that firms simply buy their inputs from the most productive suppliers without much concern for any risk involved.

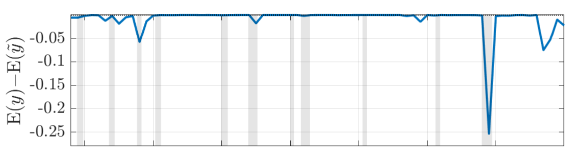
The differences between our calibrated economy and the "no uncertainty" alternatives are however larger during high-uncertainty episodes like the Great Recession. The top row of Figure 4 shows that expected log GDP in the baseline economy is about 0.25% lower in 2009 than in the alternative "as if $\Sigma_t = 0$" economy. Because of the large increase in uncertainty, firms adjust their production techniques toward safer but less productive suppliers to avoid potentially large increases in costs. The result in terms of aggregate volatility is visible in the top-right panel, where we see that log GDP is about 2.4% less volatile in 2009 in the baseline economy. Interestingly, realized log GDP, shown in the left-bottom panel, is substantially higher in the baseline economy than in the "as if $\Sigma_t = 0$" alternative. Essentially, firms took out an insurance against particularly bad TFP draws and opted for safer suppliers. When these fears were realized, this insurance policy paid off so that the baseline economy fared about 2.7% better in terms of realized log GDP compared to the alternative.

The right-bottom panel provides the same information for the "known ε_t" alternative. In this case, beliefs (μ_t, Σ_t), and in particular uncertainty, play no role in shaping the network and, from the planner's problem, the optimal network is simply the one that maximizes (realized) consumption. It follows that realized consumption (or GDP) is always larger than in the baseline model. Unsurprisingly, the difference is particularly pronounced during episodes of high uncertainty, when knowing ε_t provides a larger advantage, and reaches a high of 3% during the Great Recession.

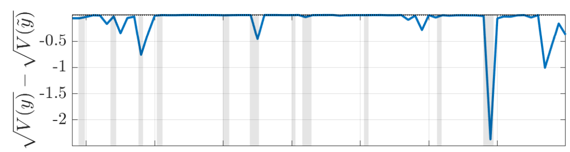
Overall, our findings suggest that, while uncertainty might have a limited impact on the economy on average, it may play a larger role in shaping the production network during high-uncertainty periods, with consequences for expected and realized GDP, as well as for welfare. Given the stylized nature of the model, these findings should be interpreted with caution. The model abstracts from other forces that might affect the production network, such as changes in demand and technological progress that would expand the set of production techniques. Similarly, the production function might not be Cobb-Douglas in reality, in which case changes in prices would affect Domar weights. We also made the implicit assumption that it takes one year (the frequency of our data) for firms to change production techniques. While this assumption might be reasonable for some sectors, it is likely that the time it takes to retool a factory varies significantly by industry, or even depending on what the new and the old techniques are. While we believe that the mechanisms that we explore in this paper would still be present in a richer model, more work would be needed to fully assess their importance.

Figure 4: The role of uncertainty in the postwar period.

First row: "as if $\Sigma_t = 0$" as the alternative



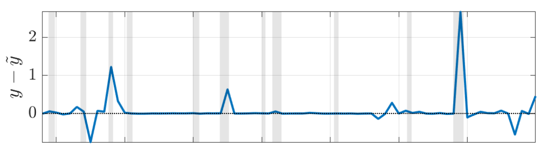
(a) Difference in expected log GDP [%]



(b) Difference in expected st. dev. of log GDP [%]

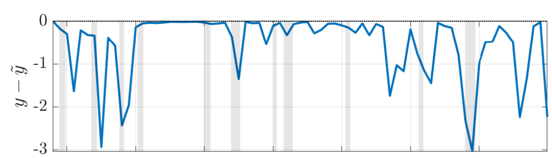
(c) Left column: "as if $\Sigma_t = 0$" as alternative

Difference in realized log GDP [%]



(d) Right column: "known ε_t" as alternative

Difference in realized log GDP [%]



9 Model-free evidence for the mechanisms

The model proposed in this paper relies on simplifying assumptions for tractability. In this section, we present additional evidence in support of the main mechanisms of the model that does not rely on this structure. Through firm-level regressions that closely follow Alfaro, Bloom, and Lin (2019) we document that 1) higher uncertainty about a firm leads to a decline in its Domar weight, and 2) network connections involving riskier suppliers are more likely to break down. We test these predictions at the firm level to take advantage of the abundance of data and of instrumental variables that are available at this level of aggregation. Supplemental Appendix E in Kopytov et al. (2024) describes the data and the instruments in detail.

9.1 Uncertainty and Domar weights

We first test the model's prediction that Domar weights decrease with uncertainty. We use annual U.S. data from 1963 to 2016 provided by Compustat. Our main variables of interest area firm's Domar weight, constructed by dividing its sales by nominal GDP, and a measure of itsstock price volatility, which we use as a proxy for uncertainty. We then regress the change in Domar weight on the change in stock price volatility. The results are presented in the first column of Table I. In column (2), we follow Alfaro et al. (2019) and address potential endogeneity concerns by instrumenting stock price volatility with industry-level exposure to ten aggregate sources of uncertainty shocks. In column (3), we use option prices to back out an implied measure of future volatility. In all cases, we find a negative and significant relationship between uncertainty and Domar weights. The effect is also economically large with a decline in Domar weight of about 18% following a doubling in firm-level volatility (roughly a 3.3 standard deviation volatility shock), according to the IV estimates. Overall, these results provide evidence that higher uncertainty leads to lower Domar weights, in line with the predictions of our theoretical model.

9.2 Uncertainty and link destruction

We conduct a similar exercise, this time at the firm-to-firm relationship level, to investigate whether higher supplier uncertainty is associated with a higher likelihood of link destruction. We proceed by combining the uncertainty data described above with data from 2003 to 2016 about firm-level supply relationships provided by Factset. We then regress a dummy variable that equals one in the last year of a relationship on the change in the supplier's stock price volatility. The results are presented in column

Table 1: Domar weights and uncertainty

	Change in Domar weight		
	(1): OLS	(2): IV	(3): IV
Δ Volatility$_{i,t-1}$	-0.058*	-0.137*	-0.218*
	(0.004)	(0.034)	(0.073)
1st moment 10IV$_{i,t-1}$	No	Yes	Yes
Type of volatility	Realized	Realized	Implied
Fixed effects	Yes	Yes	Yes
Observations	112,563	27,380	17,151
F-statistic	—	14.2	9.8

(1) of Table II. As in the last exercise, column (2) uses industry level sensitivity to aggregate shocks as instruments, and column (3) uses implied volatility from option prices as a measure of uncertainty. In all cases, we find a positive and statistically significant relationship between supplier volatility and the end of supply relationships, which is consistent with buyers moving away from riskier suppliers. The effect is also economically large with a doubling in volatility associated with a 12 percentage point increase in the likelihood that a relationship is destroyed, according to the IV estimates.

Table 2: Link destruction and supplier volatility

	Dummy for last year of supply relationship		
	(1): OLS	(2): IV	(3): IV
ΔVolatility$_{t-1}$ of supplier	0.026	0.097*	0.144
	(0.012)	(0.035)	(0.063)
1st moment 10IV$_{t-1}$ of supplier	No	Yes	Yes
Type of volatility	Realized	Realized	Implied
Fixed effects	Yes	Yes	Yes
Observations	35,629	35,620	26,195
F-statistic	—	22.9	10.4

10 Additional results related to the calibrated economy

10.1 Data

The Bureau of Economic Analysis (BEA) provides sectoral input-output tables that allow us to compute the intermediate input shares as well as the shares of final consumption expenditure accounted for by different sectors. We rely on the harmonized tables constructed by Vom Lehn and Winberry (2022) that provide consistent annual data for n = 37 sectors over the period 1948-2020.

10.2 Calibratyion procedure

The three groups of parameters that we need to calibrate are 1) the household's preferences, i.e. the consumption shares β and the risk-aversion ρ, 2) the parameters of the TFP shifter function (2), and 3) the processes for the exogenous sectoral productivity shocks, i.e. µt and Σ_t. Some of these parameters can be computed directly from the data. The other ones are estimated using a combination of indirect inference and standard time-series methods. Below, we describe the exact procedure used for each set of parameters.

Household preferences

Since the preference parameter i corresponds to the household's expenditure share of good i, we pin down its value directly from the data by averaging the consumption share of good i over time. The sectors with the largest consumption shares are "Real estate" (14%), "Retail trade" (12%) and "Health care" (11%). See Supplemental Appendix H in Kopytov et al. (2024) for a version of the calibrated economy with time-varying β's.

The relative risk aversion parameter ρ determines to what extent firms are willing to trade off higher input prices for access to more stable suppliers. The literature uses a broad range of values for ρ and it is unclear a prior i which one is best for our application. We therefore estimate ρ using a method of simulated moments (MSM) described below.

Endogenous productivity shifter

We specialize the TFP shifter function (2) to

$$\log A_i(\alpha_i) = a_i^\circ - \sum_{j=1}^{n} \kappa_{ij}(\alpha_{ij} - \alpha_{ij}^\circ)^2 - \kappa_{i0}\left(\sum_{j=1}^{n} \alpha_{ij} - \sum_{j=1}^{n} \alpha_{ij}^\circ\right)^2, \tag{69}$$

where the last term can provide a penalty from deviating from an ideal labor share. We denote by κ the matrix with typical element κ_{ij}. This functional form takes as inputs the ideal shares α_{ij}°, the actual shares α_{ijt}, the coefficients κ_{ij} and the constant a_i°. The ideal shares α_{ij}° are set to the time average of the input shares observed in the data. We set the constant a_i° equal to the average TFP of sector i. The coefficients κ_{ij}, which determine how costly it is to deviate from the ideal shares in terms of productivity, are estimated using the MSM procedure described below. Without any restrictions the matrix κ would have $n \times (n+1) = 1406$ elements. To reduce the number offree parameters to estimate, we restrict κ to be of the form $\kappa = \kappa^i \kappa^j$ where κ^i is an $n \times 1$ column vector and κ^j is an $1 \times (n+1)$ row vector. The kth element of κ^i then scales the cost for producer k of changing the share of any of its inputs, and the lth element in κ^j scales the cost of changing the share of input l for any producer. We normalize the first element in κ^i to pin down the scaleof κ^i and κ^j. The matrix κ then contains only $2n = 74$ free parameters to estimate.

Exogenous productivity process

The source of uncertainty in the model is the vector of productivity shocks $\varepsilon_t \sim \mathcal{N}(\mu_t, \Sigma_t)$. In the calibrated model, we allow μ_t and Σ_t to vary over time to account for changes in the stochastic process for ε_t over the sample period. To parameterize the evolution of μ_t and Σ_t, we first filter out the endogenous productivity shifter $A_i(\alpha_{it})$ and the normalization term $\zeta(\alpha_{it})$ from the measured sectoral TFP, $e^{\varepsilon_{it}} Ai(\alpha_{it})\zeta(\alpha_{it})$, implied by the production function (1). We then estimate the evolution of μ_t and Σ_t from the remaining component. To do so, we assume that ε_t follows a random walk with drift,

$$\varepsilon_t = \gamma + \varepsilon_{t-1} + u_t \tag{70}$$

where γ is an $n1$ vector of deterministic drifts and $u_t \sim \text{iid}\mathcal{N}(0, \Sigma_t)$ is a vector of shocks. We estimate γ by computing the average of the productivity growth rates $\Delta\varepsilon_t = \varepsilon_t - \varepsilon_{t-1}$ over time.

When making decisions in period t, firms know the past realizations of ε_t so that the conditionalmean of ε_t is given by $\mu_t = \gamma + \varepsilon_{t-1}$. The covariance Σ_t of the innovation u_t is estimated usinga rolling window that puts more weight on more recent observations to allow for time-varying uncertainty about sectoral productivity. Specifically, we estimate the covariance between sector i and j at time t by computing $\Sigma_{ijt} = \sum_{s=1}^{t-1} \phi^{t-s-1} u_{is} u_{js}$, where $0 < \phi < 1$ is a parameter that determines the relative weight of more recent observations. Its value is set to the sectoral average of the corresponding parameters of a GARCH$(1,1)$ model estimated on each sector's productivity innovation uit. In the calibrated economy, its value is $\phi = 0.47$. Note that this procedure implies that the time series for ε_t depends on the parameters of the TFP shifters. Therefore, the estimation of the stochastic process for sectoral productivity has to be done jointly with the estimation of κ.

Matching model and data moments

We use an indirect inference approach and estimate the parameters $\Theta \equiv \{\rho, \kappa\}$ by minimizing

$$\hat{\Theta} = \arg\min_{\Theta} (m(z) - m(\Theta))^T W (m(z) - m(\Theta))$$

where $m(z)$ is a vector of moments computed from the data, and $m(\Theta)$ is the vector of corresponding model-implied moments conditional on the parameters Θ. The moments that we target are the time series of the production shares α_{ijt}, normalized by their average in the data, and the demeaned time series of aggregate consumption growth, normalized by the average of its absolute value in the data. We target consumption since the stochastic discount factor of the household is central to the trade-off that firms face when choosing production techniques.

We match $n^2 \times T + T - 1$ moments with only $2n+1$ free parameters. The model is thus strongly over-identified. We use particle swarm optimization to find the global minimizer Θ (Kennedy and Eberhart, 1995). The estimated coefficient of relative risk aversion $\hat{\rho}$ is 4.27, which is similar tovalues used or estimated in the macroeconomics literature.

10.3 The calibrated economy

We want our model to fit key features of the data that relate to 1) the structure of the production network, 2) how the network responds to changes in beliefs, and 3) how this response affects macroeconomic aggregates. As we have seen earlier, the Domar weights, and how they react to changes in μ_t and Σ_t, play a central role for these mechanisms. In this section, we first describe the evolution of μ_t and Σ_t in the calibrated economy. We then report unconditional moments of the model-implied Domar weights and how they compare to the data. Finally, we look at the relationship between the Domar weights and the beliefs μ_t and Σ_t and verify that the correlations predicted by the mechanisms of the model are present in the data.

Evolution of beliefs in the data

Our estimation procedure provides a time-series for μ_t and Σ_t. To illustrate the overall evolution of beliefs over our sample period, we compute two measures that capture the aggregate impact of changes in μ_t and Σ_t. The first measure is the Domar-weighted average growth in the conditional mean of productivity, defined as

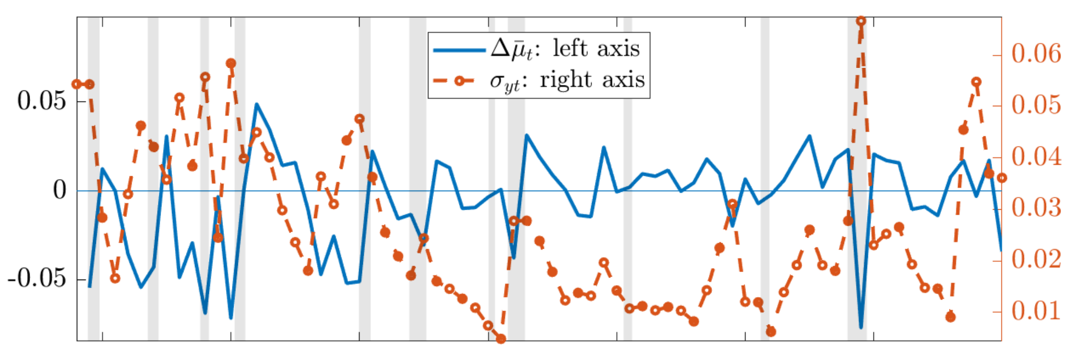
$$\Delta\bar{\mu}_t = \sum_{j=1}^{n} \omega_{jt}\Delta\mu_{jt} \tag{70}$$

We use the Domar weights ω_{jt} in this equation to properly reflect the importance of a sector for GDP, as implied by (13). The solid blue line in Figure 5 shows the evolution of $\Delta\bar{\mu}_t$ over the sample period. As expected, $\Delta\bar{\mu}_t$ tends to go below zero during NBER recessions and is positive during expansions.

To describe how aggregate uncertainty evolves in the calibrated economy, we also compute the within-period perceived standard deviation of log GDP. From (14), this can be written as

$$\sigma_{yt} = \sqrt{\mathbb{Y}[y]} = \sqrt{\omega_y^T \Sigma_t \omega_t} \tag{71}$$

Figure 5: Domar-weighted TFP and uncertainty changes



The red dashed line in Figure 5 represents the evolution of Σ_{yt} over the sample period. While uncertainty is on average relatively low, especially during the Great Moderation era, spikes are clearly visible in the earlier years and, in particular, during the Great Recession of 2007-2009.

Notes: Solid blue line: Domar-weighted average growth in the conditional mean of productivity, $\Delta\bar{\mu}_t = \sum_{j=1}^{n} \omega_{jt}\Delta\mu_{jt}$. Red dashed line: Domar-weighted conditional variance of productivity, $\Sigma_{yt} = \sqrt{\omega_t^T \Sigma_t \omega_t}$. Shaded areas represent NBER recessions.

Unconditional Domar weights

Figure 6: Sectoral Domar weights in the data and the model

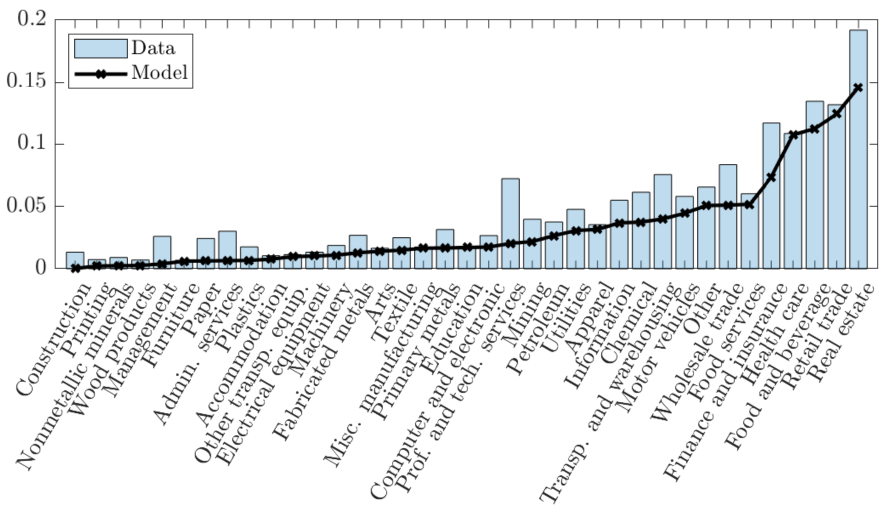


Figure 6 shows the average Domar weight of each sector in the data (blue bars) and in the model (black line). The sectors with the highest Domar weights in the data are “Real estate”, “Food and beverage”, “Retail trade”, “Finance and insurance” and “Health care”. According to our theory (Corollary 4), changes in the expected level and variance of productivity in those sectors will have the largest effects on welfare.

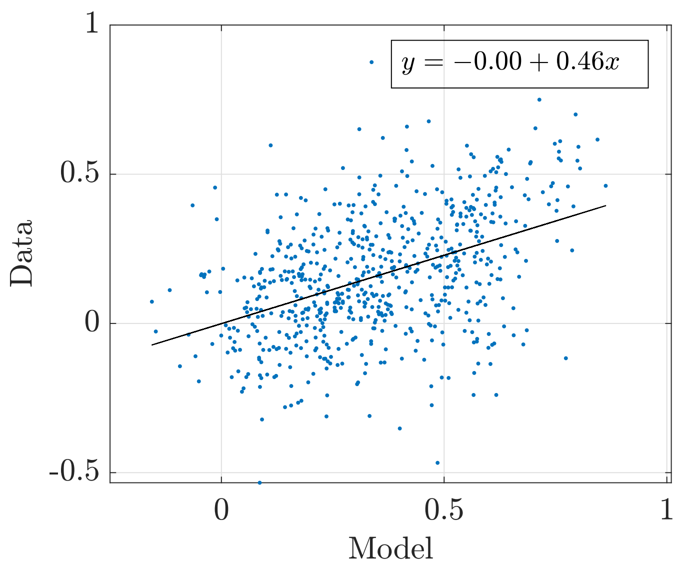
The cross-sectional correlation between the average Domar weights in the model and in the data is 0.96, so that the calibrated model fits this important feature of the production network well. However, the average Domar weight in the model (0.032) is lower than its counterpart in the data (0.047). This is because the estimation also targets aggregate consumption growth. Given the observed variation in TFP, if the model were to match the Domar weights perfectly, consumption would be too volatile compared to the data. Under our calibration, the volatility of consumption growth in the model is 2.73%, close to its data target of 2.65% (row (6) of Table IV).

The model can account for about 40% of the observed average standard deviation of the Domar weights over time, as shown in row (2) of Table IV. Row (3) also reports that the coefficient of variation of the Domar weights in the model is 0.07 compared to 0.11 in the data. Once we take into account their relative scale, the model can thus account for a sizable portion of the variation in a key moment that characterizes the production network.

Table 4: Domar weights, consumption and TFP in the model and in the data

	Statistic	**Data**	**Model**
(1)	Average Domar weight $\bar{\omega}_j$	0.047	0.032
(2)	Standard deviation $\sigma(\omega_j)$	0.0050	0.0021
(3)	Coefficient of variation $\sigma(\omega_j)/\bar{\omega}_j$	0.107	0.066
(4)	Corr (ω_{jt}, μ_{jt})	0.08	0.08
(5)	Corr $(\omega_{jt}, \Sigma_{jjt})$	-0.37	-0.31
(6)	Consumption growth volatility	2.65%	2.73%
(7)	TFP growth volatility	1.83%	2.73%

Figure 7: Cross-sector correlations in the model and in the data



Domar weights and beliefs

One of the key mechanisms of the model predicts that a decline in the expected productivity of a sector, or an increase in its variance, should lead firms to reduce the importance of that sector as an input provider, leading to a decline in its Domar weight. Proposition 2 makes this point formally for a single change in μ_i or Σ_{ii}. Of course, in the data multiple changes in μ_t and Σ_t occur at the same time, and it would be difficult to isolate the impact of a single change on the Domar weights. Instead, we look at simple cross-sector correlations between the Domar weights ω_{it} and the first (μ_{it}) and the second moments (Σ_{iit}) of sectoral TFPs, both in the data and in the model. These correlations provide a straightforward, albeit noisy, measure of the interrelations between ω_t, μ_t and Σ_t. As can be seen in rows (4) and (5) of Table IV, the predictions of the model are borne out in the data. The model is thus able to capture well the impact of beliefs on the structure of the production network.

Sectoral correlations

The model is also able to replicate features of the correlation between sectoral outputs. We focus on growth rates to accommodate different trends in the data and in the model. For each pair of sectors, we compute the correlation in their output growth in the model and in the data, and plot them in Figure 7. The model reasonably captures cross-sectoral comovements: We find that the correlation between the data- and model-implied values is 0.44. On average, sectoral outputs are positively correlated in the model and in the data, although the model correlation is somewhat weaker on average (see the first column of Table V).

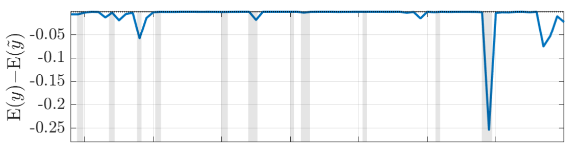
Table V also reports averages of these correlations during periods of low and high TFP growth and uncertainty growth, as measured by (70) and (71). We see that in the data these correlations are lower in good times, when TFP growth is high and uncertainty growth is low. The model is able to replicate this ranking. Intuitively, in bad times consumption is low and so the household is particularly worried about bad shocks. To avoid them, firms rely more on the most stable producers. As firms are mostly purchasing from the same sectors, sectoral outputs become more correlated.

10.4 Counterfactual exercises

Figure 8: The role of uncertainty in the postwar period

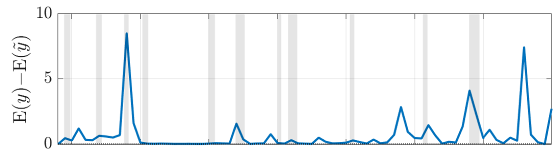
(a) Left column: the "as if $\Sigma_t = 0$" alternative

Difference in expected log GDP [%]

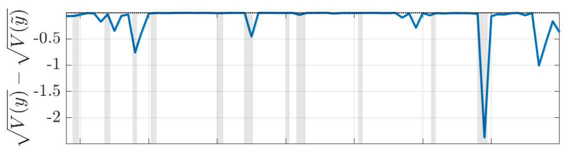


(b) Left column: the "the known ε_t" alternative

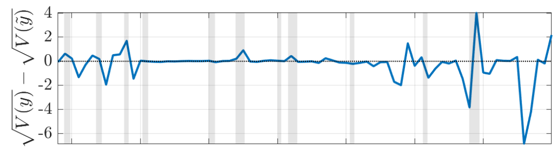
Difference in expected log GDP [%]



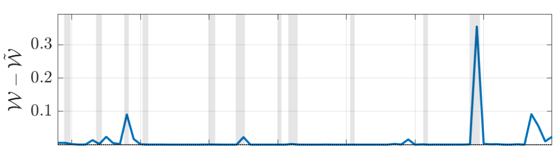
(c) Difference in expected st. dev. of log GDP [%]



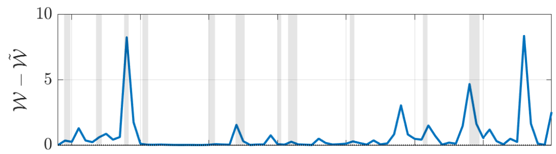
(d) Difference in expected st. dev. of log GDP [%]



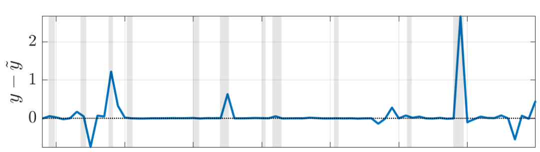
(e) Difference in expected welfare [%]



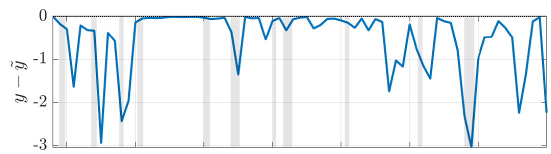
(f) Difference in expected welfare [%]



(g) Difference in realized log GDP [%]



(h) Difference in realized log GDP [%]



11 Code

11.1 Table 1

11.1.1 domar_volatility.do

```
Input Variables
1. lag_log_sales: The logarithm of the previous period's sales.
2. dhs_sale: A measure related to sales, likely based on the Davis-Haltiwanger (1992) growth definition.
3. year: The year associated with each observation.
4. permno: A unique identifier for each company.
5. gdp: Gross Domestic Product, used to calculate the Domar weight.
6. sic_3_digit: The 3-digit Standard Industrial Classification code for each company.

Output Variables
1. lag_sales: The previous period's sales, calculated as the exponential of `lag_log_sales`.
2. sales: The current period's sales, derived using the Davis-Haltiwanger growth measure.
3. domar: A weight calculated as the ratio of `sales` to `gdp`.
4. dhs_domar: The Davis-Haltiwanger growth measure for `domar`, calculated as the difference in `domar`
     from the previous period, standardized.
5. sic_2_digit: The 2-digit Standard Industrial Classification code, derived from `sic_3_digit`.

Auxiliary Code
No additional functions are called; all operations are performed using built-in Stata commands.
```

```
clear all
set more off
cls

* change working directory before running
* data from Alfaro-Bloom-Lin
use "data/Finance_Uncertainty_Multiplier/data_tables_3_4_5_6.dta", clear

gen lag_sales = exp(lag_log_sales)

* Using the definition of Davis-Haltiwanger (1992) growth measure we can back out current sales
gen sales = lag_sales *(1+0.5*dhs_sale)/(1-0.5*dhs_sale)

sort year
```

```

* Nominal GDP data from Fred
merge year using "data/nominal_gdp/gdp.dta"

drop if _merge == 2
drop _merge

gen domar = sales/gdp

tsset permno year

sort permno year

by permno: gen dhs_domar = (domar - l.domar)/(0.5*domar + 0.5 *l.domar)


*** Do the estimation
local included_instruments_1 lag_bdr lag_realized_ret lag_log_sales lag_roa lag_tangibility lag_q

* aggregate 1st moment shock controls (where dhs stands for Davis-Haltiwanger (1992) growth definition):
local included_instruments_2 lag_dhs_price_cad lag_dhs_price_euro lag_dhs_price_jpy lag_dhs_price_aud ///
       lag_dhs_price_sek lag_dhs_price_chf lag_dhs_price_gbp lag_dhs_price_epu lag_dhs_price_tyvix lag_dhs_price_oil

* 2nd moment volatility shock instruments:
local excluded_instruments lag_dhs_vol_cad lag_dhs_vol_euro lag_dhs_vol_jpy lag_dhs_vol_aud ///
       lag_dhs_vol_sek lag_dhs_vol_chf lag_dhs_vol_gbp lag_dhs_vol_epu lag_dhs_vol_tyvix lag_dhs_vol_oil



gen sic_2_digit=floor(sic_3_digit/10)
local fixed_effects year##sic_2_digit

* regressions (Table 1)
ivreghdfe dhs_domar lag_drvol, absorb(`fixed_effects') cluster(sic_3_digit)
ivreghdfe dhs_domar `included_instruments_2' (lag_drvol=`excluded_instruments'), absorb(`fixed_effects') cluster(sic_3_digit) small
ivreghdfe dhs_domar `included_instruments_2' (lag_divol=`excluded_instruments'), absorb(`fixed_effects') cluster(sic_3_digit) small
```

11.2 Table 2

11.2.1 clean_factset_data.do

```
 Input Variables
1. company_id: Company identifier in the `company_info_synth.dta` dataset.
2. country: Country associated with each company in the `company_info_synth.dta` dataset.
3. start_: Start date for the relationship in the `raw_data_synth.dta` dataset.
4. end_: End date for the relationship in the `raw_data_synth.dta` dataset.
5. source_company_id: Company identifier for the source company in the `raw_data_synth.dta` dataset.
6. target_company_id: Company identifier for the target company in the `raw_data_synth.dta` dataset.
7. source_ticker: Ticker symbol for the source company in the `raw_data_synth.dta` dataset.
8. target_ticker: Ticker symbol for the target company in the `raw_data_synth.dta` dataset.
9. rel_type: Relationship type (either "CUSTOMER" or "SUPPLIER") in the `raw_data_synth.dta` dataset.
10. source_cusip: CUSIP code for the source company in the `raw_data_synth.dta` dataset.
11. target_cusip: CUSIP code for the target company in the `raw_data_synth.dta` dataset.

 Output Variables
1. source_company_id: Renamed from `company_id` in `company_info_synth.dta` and later merged.
2. source_country: Renamed from `country` in `company_info_synth.dta`.
3. target_company_id: Renamed from `company_id` in `company_info_synth.dta` and later merged.
4. target_country: Renamed from `country` in `company_info_synth.dta`.
5. rel_type: Formatted as a string variable.
6. source_ticker: Formatted as a string variable.
7. target_ticker: Formatted as a string variable.
8. source_cusip: Converted to a 6-character CUSIP code.
9. target_cusip: Converted to a 6-character CUSIP code.
10. year_start: Year extracted from `start_` date.
11. year_end: Year extracted from `end_` date and capped at 2017.
12. year: Generated for each year the relationship is active.
13. supp: Long variable for supplier company ID.
14. cust: Long variable for customer company ID.
15. supp_cusip: CUSIP code for the supplier company.
16. cust_cusip: CUSIP code for the customer company.
17. supp_ticker: Ticker symbol for the supplier company.
18. cust_ticker: Ticker symbol for the customer company.
19. supp_country: Country for the supplier company.
20. cust_country: Country for the customer company.

```

```
 Auxiliary Code
No additional functions are called; all operations are performed using built-in Stata commands.
```

```
clear all
set more off
cls

* change working directory before running


local flag_step_one 1


if `flag_step_one'==1 {
	* Arrange the company info data for future merge
	use "raw/company_info_synth.dta", clear
	format %9.0f company_id
	drop if company_id==.
	keep company_id country
	rename company_id source_company_id
	rename country source_country
	duplicates drop
	sort source_company_id
	save company_info_source, replace

	rename source_company_id target_company_id
	rename source_country target_country
	save company_info_target, replace
}


* Now load the link database
use start_ end_ source_company_id target_company_id source_ticker target_ticker rel_type target_cusip source_cusip /*
	*/ if rel_type == "CUSTOMER" | rel_type == "SUPPLIER" using "raw/raw_data_synth.dta", clear


format %12s rel_type source_ticker target_ticker source_cusip target_cusip

```

```
destring source_company_id, replace
destring target_company_id, replace

format %9.0f source_company_id target_company_id


* First 6 digit identify the issuer
gen target_cusip_6 = substr(target_cusip, 1, 6)
drop target_cusip
rename target_cusip_6 target_cusip

gen source_cusip_6 = substr(source_cusip, 1, 6)
drop source_cusip
rename source_cusip_6 source_cusip



sort source_company_id
merge m:1 source_company_id using company_info_source
keep if _merge == 3
drop _merge


sort target_company_id
merge m:1 target_company_id using company_info_target
keep if _merge == 3
drop _merge



compress


* Create the year variables from start and end date
gen year_start = year(start_)
gen year_end = year(end_)

replace year_end = 2017 if year_end >= 2017
```

```
order year_start year_end



*create an observation for each year for which link is operating
gen dummy = _n
gen length = year_end - year_start+1
expand length
sort dummy
by dummy : gen year = year_start + _n - 1


drop dummy length year_start year_end



gen long supp = .
gen long cust = .
format %9.0f supp cust
gen supp_cusip = ""
gen cust_cusip = ""
gen supp_ticker = ""
gen cust_ticker = ""
gen supp_country = ""
gen cust_country = ""
replace supp = source_company_id if rel_type == "CUSTOMER"
replace cust = target_company_id if rel_type == "CUSTOMER"
replace supp_cusip = source_cusip if rel_type == "CUSTOMER"
replace cust_cusip = target_cusip if rel_type == "CUSTOMER"
replace supp_ticker = source_ticker if rel_type == "CUSTOMER"
replace cust_ticker = target_ticker if rel_type == "CUSTOMER"
replace supp_country = source_country if rel_type == "CUSTOMER"
replace cust_country = target_country if rel_type == "CUSTOMER"

replace cust = source_company_id if rel_type == "SUPPLIER"
replace supp = target_company_id if rel_type == "SUPPLIER"
```

```
replace cust_cusip = source_cusip if rel_type == "SUPPLIER"
replace supp_cusip = target_cusip if rel_type == "SUPPLIER"
replace cust_ticker = source_ticker if rel_type == "SUPPLIER"
replace supp_ticker = target_ticker if rel_type == "SUPPLIER"
replace cust_country = source_country if rel_type == "SUPPLIER"
replace supp_country = target_country if rel_type == "SUPPLIER"


keep supp* cust* year* start* end*
*delete repetitions
sort supp cust supp_cusip cust_cusip year
by supp cust supp_cusip cust_cusip year: gen id=_n
drop if id>1
drop id



compress

* Clean up
erase company_info_source.dta
erase company_info_target.dta

save network_factset_synth, replace
```

11.2.2 links_volatility.do

```
 Input Variables
1. permno: Unique identifier for each company in the Excel file.
2. gvkey: Global company key from the Excel file.
3. NCUSIP: CUSIP code from the Excel file.
4. TICKER: Ticker symbol from the Excel file.
5. year: Year associated with each observation in the Excel file and `network_factset_synth.dta`.
6. supp_cusip: Supplier CUSIP code from `network_factset_synth.dta`.
7. start_: Start date for the relationship in `network_factset_synth.dta`.
8. end_: End date for the relationship in `network_factset_synth.dta`.
9. cust: Customer company identifier in `network_factset_synth.dta`.
10. source_company_id: Source company identifier in `network_factset_synth.dta`.
```

```

 Output Variables
1. cusip_full: Full CUSIP code from the Excel file, renamed from `cusip`.
2. cusip: 6-character CUSIP code, derived from `cusip_full`.
3. year_start: Year extracted from `start_` date.
4. year_end: Year extracted from `end_` date, capped at 2017.
5. year: Year for each observation where the relationship is active.
6. supp_permno: Supplier company identifier, renamed from `permno`.
7. cust_id: Customer company identifier, renamed from `cust`.
8. supp_id: Supplier company identifier, renamed from `supp`.
9. max_year_cust: Maximum year for customer observations.
10. max_year_supp: Maximum year for supplier observations.
11. max_year: Maximum year for each relationship spell.
12. min_year: Minimum year for each relationship spell.
13. nb_year: Number of years each relationship is active.
14. last_year_link: Indicator for the last year the link is active.
15. total_last_link: Total number of last year links per year.
16. total_obs: Total number of observations per year.
17. supp_sic_2_digit: 2-digit SIC code for suppliers, derived from `supp_sic_3_digit`.
18. supp_sic_1_digit: 1-digit SIC code for suppliers, derived from `supp_sic_2_digit`.

 Auxiliary Code
No additional functions are called; all operations are performed using built-in Stata commands.
```

```
clear all
set more off
cls

* change working directory before running


local link_flag=1

*cusip to permno match
if `link_flag'==1 {
	import excel "data/Finance_Uncertainty_Multiplier/Alfaro_Bloom_Lin_FUM_IVS.xlsx", sheet("AlfaroBL_firm_uncertainty_IVs") firstrow clear

	keep permno gvkey NCUSIP TICKER year
```

```
        rename NCUSIP cusip
        rename TICKER ticker

        gen cusip_6 = substr(cusip, 1, 6)
        rename cusip cusip_full
        rename cusip_6 cusip
        save TEMP, replace

}

*factset data
use "data/Factset_Revere/network_factset_synth.dta", clear
drop if supp_cusip==""


gen cusip = supp_cusip
sort cusip year

merge m:1 cusip year using TEMP
keep if _merge==3
drop _merge

bys cusip cust year: gen id=_n
drop if id>1
drop id


* Merge with Alfaro-Bloom-Lin's data
sort permno year
merge m:1 permno year using "data/Finance_Uncertainty_Multiplier/data_tables_3_4_5_6.dta"
keep if _merge==3
rename permno supp_permno
drop start_ end_ _merge



local temp_list "lag_bdr lag_q lag_realized_ret lag_log_sales lag_roa lag_tangibility sic_3_digit capx_
    lag_ppent lag_dhs_vol_oil lag_dhs_price_oil lag_dhs_vol_cad lag_dhs_price_cad lag_dhs_vol_euro lag_
    dhs_price_euro lag_dhs_vol_jpy lag_dhs_price_jpy lag_dhs_vol_aud lag_dhs_price_aud lag_dhs_vol_sek
```

```
    lag_dhs_price_sek lag_dhs_vol_chf lag_dhs_price_chf lag_dhs_vol_gbp lag_dhs_price_gbp lag_dhs_vol_epu
     lag_dhs_price_epu lag_dhs_vol_tyvix lag_dhs_price_tyvix lag_drvol lag_divol dhs_xsga_xrd dhs_emp dhs
    _cogs dhs_sale dhs_debt_total dhs_payout dhs_che"
foreach vartemp of local temp_list {
       rename `vartemp' supp_`vartemp'
}


gen supp_id=supp_permno
gen cust_id=cust
drop supp cust
rename supp_id supp
rename cust_id cust



* Figure out if firms exit the dataset
bysort cust: egen max_year_cust = max(year)
bysort supp: egen max_year_supp = max(year)

* Figure out the relationship spells
sort cust supp year
by cust supp: egen max_year = max(year)
by cust supp: egen min_year = min(year)
by cust supp: egen nb_year = count(year)

* Keep firms that had a continuous link for at least 5 years
drop if nb_year < 5
drop if nb_year < max_year-min_year+1 /* Drop non-continuous spells */

gen last_year_link = 0
replace last_year_link = 1 if year == max_year

bys year: egen total_last_link=total(last_year_link)
by year: gen total_obs=_N

drop if total_last_link==0
drop if total_last_link==total_obs

```

```
sort cust supp year


*** Do the estimation
local included_instruments_1_supp supp_lag_bdr supp_lag_realized_ret supp_lag_log_sales supp_lag_roa supp_lag_tangibility supp_lag_q

* aggregate 1st moment shock controls (where dhs stands for Davis-Haltiwanger (1992) growth shock)
local included_instruments_2_supp supp_lag_dhs_price_cad supp_lag_dhs_price_euro supp_lag_dhs_price_jpy supp_lag_dhs_price_aud ///
    supp_lag_dhs_price_sek supp_lag_dhs_price_chf supp_lag_dhs_price_gbp supp_lag_dhs_price_epu supp_lag_dhs_price_tyvix supp_lag_dhs_price_oil


* 2nd moment volatility shock instruments:
local excluded_instruments_supp supp_lag_dhs_vol_cad supp_lag_dhs_vol_euro supp_lag_dhs_vol_jpy supp_lag_dhs_vol_aud ///
    supp_lag_dhs_vol_sek supp_lag_dhs_vol_chf supp_lag_dhs_vol_gbp supp_lag_dhs_vol_epu supp_lag_dhs_vol_tyvix supp_lag_dhs_vol_oil


gen supp_sic_2_digit=floor(supp_sic_3_digit/10)
gen supp_sic_1_digit=floor(supp_sic_2_digit/10)
local fixed_effects year##supp_sic_2_digit##cust


* regressions (Table 2)
ivreghdfe last_year_link supp_lag_drvol, absorb(`fixed_effects') cluster(cust supp_sic_3_digit)
ivreghdfe last_year_link `included_instruments_2_supp' (supp_lag_drvol=`excluded_instruments_supp'), absorb(`fixed_effects') cluster(cust supp_sic_3_digit) small
ivreghdfe last_year_link `included_instruments_2_supp' (supp_lag_divol=`excluded_instruments_supp'), absorb(`fixed_effects') cluster(cust supp_sic_3_digit) small


 erase TEMP.dta
```

11.3 Figure 1

11.3.1 main.m

```
 Input Variables
1. param.A_i_o: TFP under ideal shares.
2. param.alpha_o: Ideal shares for three goods.
3. param.kappa: Penalty from deviating from the ideal shares.
4. param.psi1: Parameter related to the firm's cost function.
5. param.psi2: Parameter related to the firm's cost function.
6. param.Vlambda: Parameter for variance of prices.
7. param.Cplambda: Parameter for mean of prices.
8. n_Ep: Number of points in the grid for the mean of price for good 1.
9. n_Vp: Number of points in the grid for the variance of price for good 1.
10. Ep1_grid: Grid of mean prices for good 1.
11. Vp1_grid: Grid of variance prices for good 1.
12. Ep1: Mean price for good 1 (unchanged).
13. Vp1: Variance of price for good 1 (unchanged).
14. Ep2: Mean price for good 2.
15. Vp2: Variance of price for good 2.
16. Ep3: Mean price for good 3.
17. Vp3: Variance of price for good 3.
18. alphai1_Ep1: Input shares for good 1 when changing mean prices.
19. alphai2_Ep1: Input shares for good 2 when changing mean prices.
20. alphai3_Ep1: Input shares for good 3 when changing mean prices.
21. alphai1_Vp1: Input shares for good 1 when changing variance of prices.
22. alphai2_Vp1: Input shares for good 2 when changing variance of prices.
23. alphai3_Vp1: Input shares for good 3 when changing variance of prices.

 Output Variables
1. alphai1_Ep1: Optimized input shares for good 1 when mean prices vary.
2. alphai2_Ep1: Optimized input shares for good 2 when mean prices vary.
3. alphai3_Ep1: Optimized input shares for good 3 when mean prices vary.
4. alphai1_Vp1: Optimized input shares for good 1 when variance of prices vary.
5. alphai2_Vp1: Optimized input shares for good 2 when variance of prices vary.
6. alphai3_Vp1: Optimized input shares for good 3 when variance of prices vary.

 Auxiliary Code
```

```
No additional functions are called; all operations are performed using built-in MATLAB commands and the
    custom function `to_minimize` which is used to solve the optimization problem.
```

```
%% Code to illustrate the impact of uncertainty in prices on sourcing decisions

% We are solving the problem of a firm i that is facing random prices for three goods
% We vary the moments of the prices for good 1


clear;

param.A_i_o = 0;                  % TFP under ideal shares
param.alpha_o = [1/3;1/3;1/3;];    % Ideal shares
param.kappa = [1/10;1/10;1/10];     % Penalty from deviating from ideal
param.psi1 = 1;
param.psi2 = 1;

% See paper equation for parameters

param.Vlambda = 1;
param.Cplambda = 1;

n_Ep = 50;
n_Vp = 50;

% Grid for mean and variance of good i
Ep1_grid = linspace(-0.5,0.5,n_Ep)'; % Mean of price for good 1
Vp1_grid = linspace(0,0.1,n_Vp)'; % Variance of prices for good 1

% Mean and variance for firm 1 when not changed
Ep1 = 0.00;
Vp1 = 0.0;

% Mean and variance for firm 2
Ep2 = -0.05;
Vp2 = 0.1;

% Mean and variance for firm 3
Ep3 = 0.05;
```

```
Vp3 = 0.1;

alphai1_Ep1 = zeros(n_Ep,1);
alphai2_Ep1 = zeros(n_Ep,1);
alphai3_Ep1 = zeros(n_Ep,1);

alphai1_Vp1 = zeros(n_Vp,1);
alphai2_Vp1 = zeros(n_Vp,1);
alphai3_Vp1 = zeros(n_Vp,1);

% Change EP1
for i_Ep = 1:n_Ep

    Ep = [Ep1_grid(i_Ep);Ep2;Ep3];
    Vp = [Vp1;Vp2;Vp3];

    f = @(x)to_minimize(x,Ep,Vp,param);
    [alpha_star,fval] = fmincon(f,[1/4,1/4,1/4],[1 1 1],1,[],[],[0;0;0],[1;1;1]);

    alphai1_Ep1(i_Ep) = alpha_star(1);
    alphai2_Ep1(i_Ep) = alpha_star(2);
    alphai3_Ep1(i_Ep) = alpha_star(3);
end


% Change VP1
for i_Vp = 1:n_Vp

    Ep = [Ep1;Ep2;Ep3];
    Vp = [Vp1_grid(i_Vp);Vp2;Vp3];

    f = @(x)to_minimize(x,Ep,Vp,param);
    [alpha_star,fval] = fmincon(f,[1/4,1/4,1/4],[1 1 1],1,[],[],[0;0;0],[1;1;1]);

    alphai1_Vp1(i_Vp) = alpha_star(1);
    alphai2_Vp1(i_Vp) = alpha_star(2);
    alphai3_Vp1(i_Vp) = alpha_star(3);
end

```

```
subplot(2,3,1);
plot(Ep1_grid,alphai1_Ep1);
ylim([0 1]);
title('\alpha_1 when chaning Ep1')

subplot(2,3,2);
plot(Ep1_grid,alphai2_Ep1);
ylim([0 1]);
title('\alpha_2 when chaning Ep1')

subplot(2,3,3);
plot(Ep1_grid,alphai3_Ep1);
ylim([0 1]);
title('\alpha_3 when chaning Ep1')

subplot(2,3,4);
plot(Vp1_grid,alphai1_Vp1);
ylim([0 1]);
title('\alpha_1 when chaning Vp1')

subplot(2,3,5);
plot(Vp1_grid,alphai2_Vp1);
ylim([0 1]);
title('\alpha_2 when chaning Vp1')

subplot(2,3,6);
plot(Vp1_grid,alphai3_Vp1);
ylim([0 1]);
title('\alpha_3 when chaning Vp1')

writematrix([Ep1_grid Vp1_grid alphai1_Ep1 alphai2_Ep1 alphai3_Ep1 alphai1_Vp1 alphai2_Vp1 alphai3_Vp1],"
    output.txt",'Delimiter','tab');


% Plot the results
figure;
subplot(1,2,1);
plot(Ep1_grid,alphai1_Ep1,'b-', 'DisplayName', '$\alpha_{41}$ steel', 'LineWidth', 2); hold on;
plot(Ep1_grid,alphai2_Ep1,'r--', 'DisplayName', '$\alpha_{42}$ steel machinery', 'LineWidth', 2);
```

```
plot(Ep1_grid,alphai3_Ep1,'g-.', 'DisplayName', '$\alpha_{43}$ carbon fiber', 'LineWidth', 2);
ylim([0 1]);
xlabel('Expected log price of steel $E[p_1]$', 'Interpreter', 'latex');
ylabel('Input shares', 'Interpreter', 'latex');
title('Impact of $E[p_1]$ on input shares', 'Interpreter', 'latex');
legend('show', 'Interpreter', 'latex');

subplot(1,2,2);
plot(Vp1_grid,alphai1_Vp1,'b-', 'DisplayName', '$\alpha_{41}$ steel', 'LineWidth', 2); hold on;
plot(Vp1_grid,alphai2_Vp1,'r--', 'DisplayName', '$\alpha_{42}$ steel machinery', 'LineWidth', 2);
plot(Vp1_grid,alphai3_Vp1,'g-.', 'DisplayName', '$\alpha_{43}$ carbon fiber', 'LineWidth', 2);
ylim([0 1]);
xlabel('Volatility of the log price of steel $V[p_1]$', 'Interpreter', 'latex');
ylabel('Input shares', 'Interpreter', 'latex');
title('Impact of $V[p_1]$ on input shares', 'Interpreter', 'latex');
legend('show', 'Interpreter', 'latex');
```

11.3.2 to_minimize.m

```
 Input Variables
1. a: Input shares vector for the three goods.
2. Ep: Expected (mean) prices vector for the three goods.
3. Vp: Variance of prices vector for the three goods.
4. param.alpha_o: Ideal shares for the three goods.
5. param.kappa: Penalty parameters for deviating from the ideal shares.
6. param.Vlambda: Variance parameter for price uncertainty.
7. param.Cplambda: Coefficient parameter for price uncertainty.
8. param.psi1: Parameter for the cost function related to the difference in input shares.
9. param.psi2: Parameter for the cost function related to the total deviation from ideal shares.

 Output Variable
1. output: The value of the objective function to be minimized, which represents the costs and
    adjustments associated with the input shares `a` given the expected prices `Ep` and price variances `
    Vp`.

 Auxiliary Code
No additional functions are called; all operations are performed using built-in MATLAB commands and the
    provided parameters. The custom function `to_minimize` calculates the objective function value based
```

```
    on the input shares, expected prices, price variances, and the given parameters.

 Explanation of the Function
1. a0: Extracts the ideal shares from the `param` structure.
2. kappa: Extracts the penalty parameters from the `param` structure.
3. Vlambda: Extracts the variance parameter from the `param` structure.
4. Cplambda: Extracts the coefficient parameter from the `param` structure.
5. psi1: Extracts the first cost function parameter from the `param` structure.
6. psi2: Extracts the second cost function parameter from the `param` structure.
7. R1, R2, R3: Calculates the adjusted prices for the three goods based on their expected prices, price
    variances, and the uncertainty parameters.
8. output: Calculates the objective function value based on the penalties for deviating from ideal shares
    , the costs associated with differences in input shares, and the total costs associated with the
    input shares and adjusted prices.
```

```
function output = to_minimize(a,Ep,Vp,param)
   % Profits under input shares alphas for firm i

   a0 = param.alpha_o;    % Ideal shares
   kappa = param.kappa;       % Penalty from deviating from ideal

   Vlambda = param.Vlambda;
   Cplambda= param.Cplambda;

   psi1 = param.psi1;
   psi2 = param.psi2;

   R1 = Ep(1) + Cplambda*sqrt(Vlambda)*sqrt(Vp(1));
   R2 = Ep(2) + Cplambda*sqrt(Vlambda)*sqrt(Vp(2));
   R3 = Ep(3) + Cplambda*sqrt(Vlambda)*sqrt(Vp(3));

   output = kappa(1)*(a(1)-a0(1))^2+kappa(2)*(a(2)-a0(2))^2+kappa(3)*(a(3)-a0(3))^2 +...
      psi1*(a(1)-a(2))^2 + psi2*(a(1)+a(3)-a0(1)-a0(3))^2 + ...
      a(1)*R1 + a(2)*R2 + a(3)*R3;

end
```

11.4 Figure 2

11.4.1 main.m

```
Input Variables
n: Number of goods.
mu: Mean vector for random variables.
mu_high: High mean value for the fourth good.
sigma: Covariance matrix for random variables.
beta: Weight vector for goods.
rho: Risk aversion parameter.
kappa: Penalty matrix for deviating from ideal shares.
kappa_inf: Infinite penalty value.
kappa_bar: Zero penalty value.
LS_min: Minimum labor share allowed.
alpha_bar: Ideal share matrix for each good and share.
alpha_inf: Infinite ideal share value.
alpha_zer: Zero ideal share value.
param: Structure containing all parameters.

Output Variables
alpha_star_1: Equilibrium input shares with initial sigma.
alpha_star_2: Equilibrium input shares with updated sigma.

Auxiliary Code
compute_eq: Custom function called to compute the equilibrium input shares given the parameters.
```

```
% Solve the model and print some moments

clear all

% Set random seed for reproducibility
rng('default')

% Number of goods
n = 7;
% Mean vector for random variables
mu = zeros(n,1);
mu_high= 0.1;
```

```
mu(4) = mu_high;
% Covariance matrix for random variables
sigma = eye(n,n)/10;
sigma(4,4) = 1;
% Weight vector for goods
beta = ones(n,1)/n;
% Risk aversion parameter
rho = 2;

% Penalty matrix for deviating from ideal shares
kappa = zeros(7,8);
kappa_inf = 1e4;
kappa_bar = 0;

% Minimum labor share allowed
LS_min = 0.00;

% Define penalty parameters for each good and share
kappa(1,:) = [kappa_inf kappa_bar kappa_inf kappa_inf kappa_bar kappa_inf kappa_inf kappa_inf];
kappa(2,:) = [kappa_inf kappa_inf kappa_bar kappa_inf kappa_inf kappa_bar kappa_inf kappa_inf];
kappa(3,:) = [kappa_inf kappa_inf kappa_inf kappa_bar kappa_inf kappa_inf kappa_bar kappa_inf];
kappa(4,:) = [kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_bar];
kappa(5,:) = [kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_bar];
kappa(6,:) = [kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_bar];
kappa(7,:) = [kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_inf kappa_bar];

% Ideal share matrix for each good and share
alpha_bar = zeros(7,8);
alpha_inf = 0;
alpha_zer = 1/2;

% Define ideal shares for each good and share
alpha_bar(1,:) = [alpha_inf alpha_zer alpha_inf alpha_inf alpha_zer alpha_inf alpha_inf 1-LS_min];
alpha_bar(2,:) = [alpha_inf alpha_inf alpha_zer alpha_inf alpha_inf alpha_zer alpha_inf 1-LS_min];
alpha_bar(3,:) = [alpha_inf alpha_inf alpha_inf alpha_zer alpha_inf alpha_inf alpha_zer 1-LS_min];
alpha_bar(4,:) = [alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf LS_min];
alpha_bar(5,:) = [alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf LS_min];
alpha_bar(6,:) = [alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf LS_min];
alpha_bar(7,:) = [alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf alpha_inf LS_min];
```

```

% Set parameters in a structure
param.n = n;
param.mu = mu;
param.sigma = sigma;
param.beta = beta;
param.rho = rho;
param.kappa = kappa;
param.alpha_bar = alpha_bar;
param.A_i_o = zeros(n,1);
param.LS_min = LS_min;

% Compute the equilibrium with initial sigma
alpha_star_1 = compute_eq(param);

% Update sigma and recompute the equilibrium
sigma(4,4) = 0;
param.sigma = sigma;
alpha_star_2 = compute_eq(param);

% Display results
disp('alpha_1 = ')
disp(alpha_star_1)
disp('alpha_2 = ')
disp(alpha_star_2)
```

11.4.2 compute_eq.m

```
Input Variables
param: Structure containing all parameters including n (number of goods), initial expected benefit matrix
    , and other relevant parameters.

Output Variables
alpha_star: Equilibrium input shares for each firm after convergence.

Auxiliary Code
solve_firm_problem: Custom function called within the loop to solve the optimization problem for each
    firm, given the current input shares. This function computes the optimal input shares for each firm.
```

```
function alpha_star = compute_eq(param)

% Compute the equilibrium

% Number of goods
n = param.n;

% Initial expected benefit of each firm's input shares
% First index firm, second index input
alpha_star = ones(n,n) * 1 / (n + 4);

% Convergence flag and iteration parameters
has_converged = false;
iter = 0;
iter_max = 1000;
tol = 1e-8;

% Iterate until convergence or maximum iterations reached
while has_converged == false && iter < iter_max
    iter = iter + 1;
    alpha_star_new = zeros(param.n, param.n);

    % Solve each firm's problem given the current input shares
    for i = 1:param.n
        alpha_star_new(i, :) = solve_firm_problem(param, i, alpha_star);
    end

    % Compute the maximum difference between old and new input shares
    max_diff = max(abs(alpha_star_new - alpha_star), [], 'all');

    % Check for convergence
    if max_diff < tol
        has_converged = true;
    else
        alpha_star = alpha_star_new;
    end
end

```

```
% Raise an error if no convergence
if iter >= iter_max
    error("No convergence")
end

end
```

11.4.3 solve_firm_problem.m

```
Input Variables
param: Structure containing all model parameters including mu, sigma, n, rho, LS_min, kappa, alpha_bar,
    and beta.
i_firm: Index of the firm for which the problem is being solved.
alpha_star: Current input shares for all firms.

Output Variables
alpha_chosen: Optimized input shares for the given firm.

Auxiliary Code
a_alpha_star: Custom function called to compute the equilibrium TFP of the firms given the current input
    shares and parameters.
quadprog: MATLAB built-in function used to solve the quadratic programming problem.
```

```
function [alpha_chosen] = solve_firm_problem(param, i_firm, alpha_star)
% Solve the problem of the firm using quadratic solver
% See app.lyx for notation

% The matrix in the quadratic optimization is often slightly (1e-15) not
% symmetric...
warning('off','optim:quadprog:HessianNotSym')

% Extract parameters
mu = param.mu;
sigma = param.sigma;
n = param.n;
rho = param.rho;
LS_min = param.LS_min;
kappa = param.kappa;
```

```
alpha_bar = param.alpha_bar;
beta = param.beta;

% Compute the matrix associated with the TFP b
B_bar = 2 * (kappa(i_firm, 1:n) .* alpha_bar(i_firm, 1:n) + kappa(i_firm, n+1) * alpha_bar(i_firm, n+1));
B_bar = B_bar';
A_bar = -ones(n, n) * kappa(i_firm, n+1);
A_bar(logical(eye(n))) = A_bar(logical(eye(n))) - kappa(i_firm, 1:n)';

% Adjust mu for quadratic optimization
mu_tilde = mu;

% Compute the equilibrium TFP of the firms
a_star = a_alpha_star(alpha_star, param);

% Compute the Leontief inverse
L = inv(eye(n, n) - alpha_star);
one_i = zeros(n, 1);
one_i(i_firm) = 1;

% The linear part of the quadratic equation
f = -(B_bar + L * (mu_tilde + a_star - sigma * (one_i - L' * (one_i + (1 - rho) * beta))));

% The quadratic part of the equation
H = 2 * (1/2 * L * sigma * L' - A_bar);

% Constraints for the optimization
A_const = ones(1, n);
b_const = (1 - LS_min);
options = optimoptions('quadprog', 'Display', 'off');

% Solve the quadratic programming problem
alpha_chosen = quadprog(H, f, A_const, b_const, [], [], zeros(n, 1), ones(n, 1), [], options);

end
```

11.4.4 a_alpha_star.m

```
Input Variables
alpha_star: Current input shares for all firms.
param: Structure containing all model parameters including n, kappa, and alpha_bar.

Output Variables
a_star: Equilibrium TFP term for each firm.

Auxiliary Code
No additional functions are called; all operations are performed using built-in MATLAB commands and the
    provided parameters.
```

```
function [a_star] = a_alpha_star(alpha_star, param)
% Compute a(alpha_star), the equilibrium TFP term coming from the input choice

% Extract parameters
n = param.n;
kappa = param.kappa;
alpha_bar = param.alpha_bar;

% Initialize the equilibrium TFP term
a_star = zeros(n, 1);

% Compute the TFP term for each firm
for j = 1:n
    % Compute B_bar for firm j
    B_bar_j = 2 * (kappa(j, 1:n) .* alpha_bar(j, 1:n) + kappa(j, n+1) * alpha_bar(j, n+1));
    B_bar_j = B_bar_j';
    % Compute A_bar for firm j
    A_bar_j = -ones(n, n) * kappa(j, n+1);
    A_bar_j(logical(eye(n))) = A_bar_j(logical(eye(n))) - kappa(j, 1:n)';
    % Compute C_bar for firm j
    C_bar_j = -(kappa(j, n+1) * (alpha_bar(j, n+1))^2 + sum(kappa(j, 1:n) .* alpha_bar(j, 1:n).^2));
    % Compute the temporary alpha vector for firm j
    alpha_temp = alpha_star(j, :)';
    % Compute the equilibrium TFP term for firm j
    a_star(j) = alpha_temp' * B_bar_j + alpha_temp' * A_bar_j * alpha_temp + C_bar_j;
end

```

```
end
```

11.5 Figure 3

11.5.1 main.m

```
Input Variables
example_mean_flag: Flag to determine if example with varying mean should be used.
example_sigma_flag: Flag to determine if example with varying variance should be used.
save_flag: Flag to determine if figures should be saved.
n: Number of firms.
N: Number of points in the grid for mean and variance of shocks.
mu_arr: Array of mean values for the shocks.
sigma_arr: Array of variance values for the shocks.
param.mu: Mean vector for shocks.
param.sigma: Covariance matrix for shocks.
param.n: Number of firms.
param.rho: CRRA risk aversion parameter.
param.H_inv: Inverse of the Hessian matrix.
param.alpha_bar: Ideal share matrix for each good and share.
param.LS_min: Minimum labor share allowed.
param.beta: Weight vector for firms.

Output Variables
E_log_C_arr: Array of expected log consumption for each grid point.
V_log_C_arr: Array of variance of log consumption for each grid point.
welfare_mu_arr: Array of welfare for each grid point with varying mean.
mean_log_P_arr: Array of mean log prices for each firm.
alpha_star_arr: Array of equilibrium input shares for each firm.
E_log_C_fixed_network_arr: Array of expected log consumption for fixed network.
V_log_C_fixed_network_arr: Array of variance of log consumption for fixed network.
domar: Array of Domar weights for each firm.
global_inv_H: Array of global inverse Hessian matrices.
H_inv: Inverse Hessian matrix for each firm.
E_log_C_deriv: Derivative of expected log consumption.
V_log_C_deriv: Derivative of variance of log consumption.
domar_deriv: Derivative of Domar weights.
welfare_arr: Array of welfare for each grid point.
```

```
welfare_fixed_network_arr: Array of welfare for fixed network.

Auxiliary Code
compute_eq: Custom function called to compute the equilibrium input shares given the parameters and
    initial input shares.
compute_moments: Custom function called to compute moments of the distribution such as expected log
    consumption, variance, mean log prices, etc.
exportgraphics: MATLAB built-in function used to save the figures to files.
```

```
% Initialize the environment
clear variables
example_mean_flag = 1;
example_sigma_flag = 0;
save_flag = 1;

% Number of firms
n = 5;

% Define the range for mean and variance of shocks
N = 201;
mu_arr = linspace(-0.08, 0.1, N);
sigma_arr = linspace(0.02, 0.2, N);

% Initialize arrays to store results
E_log_C_arr = zeros(N, 1);
V_log_C_arr = zeros(N, 1);
welfare_mu_arr = zeros(N, 1);

mean_log_P_arr = zeros(n, N);
alpha_star_arr = zeros(n, n, N);

E_log_C_fixed_network_arr = zeros(N, 1);
V_log_C_fixed_network_arr = zeros(N, 1);

domar = zeros(n, N);
global_inv_H = zeros(n, n, N);

H_inv = zeros(n, n, n);
E_log_C_deriv = zeros(N, 1);
```

```
V_log_C_deriv = zeros(N, 1);
domar_deriv = zeros(n, N);

for ii = 1:N
    % Example 1: growing mu and reducing output
    if example_mean_flag == 1
        mu = [0.1; 0.1; 0.1; 0.1; mu_arr(ii)];
        sigma = diag([0.2 0.2 0.2 0.2 sigma_arr(1)]);
        example_sigma_flag = 0;
    end

    % Example 2: growing sigma and increasing output
    if example_sigma_flag == 1
        mu = [0.1; 0.1; 0.1; 0.1; mu_arr(1)];
        sigma = diag([0.2 0.2 0.2 0.2 sigma_arr(ii)]);
    end

    factor = 256.4;
    if example_mean_flag == 1
        kappa_subst = 0.019; % if positive, then substitutes
    elseif example_sigma_flag == 1
        kappa_subst = -0.019;
    end

    kappa_large = 0.02;
    H_inv(:, :, 1) = [-kappa_large 0 0 0 0;
                      0 -kappa_large 0 0 0;
                      0 0 -kappa_large 0 0;
                      0 0 0 -kappa_large*factor kappa_subst*factor;
                      0 0 0 kappa_subst*factor -kappa_large*factor];
    H_inv(:, :, 2) = H_inv(:, :, 1);
    H_inv(:, :, 3) = H_inv(:, :, 1);
    H_inv(:, :, 4) = diag([-kappa_large -kappa_large -kappa_large -kappa_large -kappa_large]);
    H_inv(:, :, 5) = diag([-kappa_large -kappa_large -kappa_large -kappa_large -kappa_large]);

    alpha_small = 0.01 * 0;
    alpha_bar = [alpha_small alpha_small alpha_small 0.25 0.25;
                 alpha_small alpha_small alpha_small 0.25 0.25;
                 alpha_small alpha_small alpha_small 0.25 0.25;
```

```
            alpha_small alpha_small alpha_small alpha_small alpha_small;
            alpha_small alpha_small alpha_small alpha_small alpha_small];

    beta = [1 1 1 0.001 0.001]';
    beta = beta / sum(beta);

    LS_min = 0.01; % Minimum labor share allowed

    rho = 2.5; % CRRA risk aversion

    param.mu = mu;
    param.sigma = sigma;
    param.n = n;
    param.rho = rho;
    param.H_inv = H_inv;
    param.alpha_bar = alpha_bar;
    param.LS_min = LS_min;
    param.beta = beta;

    % Compute the equilibrium
    if ii > 1
       alpha_star_init = alpha_star_arr(:, :, ii-1);
    else
       alpha_star_init = ones(n, n) * 1 / (n + 10);
    end
    alpha_star = compute_eq(param, alpha_star_init);
    L = inv(eye(n, n) - alpha_star);
    domar(:, ii) = L' * beta;
    aux = zeros(n, n);
    for jj = 1:n
       aux = aux + domar(jj, ii) * H_inv(:, :, jj);
    end
    global_inv_H(:, :, ii) = inv((eye(n) - alpha_star) * inv(aux) * (eye(n) - alpha_star)' - (param.rho - 1) * sigma);

    [E_log_C, V_log_C, mean_LS, mean_log_P, covar_log_P] = compute_moments(alpha_star, param);

    mean_log_P_arr(:, ii) = mean_log_P;
    E_log_C_arr(ii) = E_log_C;
```

```
    V_log_C_arr(ii) = V_log_C;

    alpha_star_arr(:, :, ii) = alpha_star;

    [E_log_C_fixed_network_arr(ii), V_log_C_fixed_network_arr(ii)] = compute_moments(alpha_star_arr(:, :, 1), param);

    if example_mean_flag == 1
        E_log_C_deriv(ii) = domar(5, ii) - (param.rho - 1) * domar(:, ii)' * sigma * global_inv_H(:, :, ii) * [0; 0; 0; 0; 1];
        V_log_C_deriv(ii) = -2 * domar(:, ii)' * sigma * global_inv_H(:, :, ii) * [0; 0; 0; 0; 1];
        domar_deriv(:, ii) = -global_inv_H(:, :, ii) * [0; 0; 0; 0; 1];
    end
    if example_sigma_flag == 1
        E_log_C_deriv(ii) = -(param.rho - 1) * domar(:, ii)' * sigma * global_inv_H(:, :, ii) * [0; 0; 0; 0; 1] * ((1 - param.rho) * domar(5, ii));
        V_log_C_deriv(ii) = domar(5, ii)^2 - 2 * domar(:, ii)' * sigma * global_inv_H(:, :, ii) * [0; 0; 0; 0; 1] * ((1 - param.rho) * domar(5, ii));
        domar_deriv(:, ii) = -global_inv_H(:, :, ii) * [0; 0; 0; 0; 1] * ((1 - param.rho) * domar(5, ii));
    end
end

welfare_arr = E_log_C_arr + 1/2 * (1 - rho) * V_log_C_arr;
welfare_fixed_network_arr = E_log_C_fixed_network_arr + 1/2 * (1 - rho) * V_log_C_fixed_network_arr;

% Plot results if figure_flag is set
figure_flag = 1;

if figure_flag == 1
    if example_mean_flag == 1
        xx = [200, 400, 500, 400];
        close(figure(32))
        figure(32)

        colors = get(gca, 'ColorOrder');
        set(gcf, 'Position', xx)
        set(gca, 'TickLabelInterpreter', 'latex');
        box on
        grid on
```

```
    hold on

    x_A = mu_arr(1);
    y_A = E_log_C_arr(1);
    plot(mu_arr, y_A * ones(size(mu_arr)), 'color', 'k', 'linewidth', 0.5)

    h1 = plot(mu_arr, E_log_C_arr, 'color', colors(1, :), 'linewidth', 2);
    h2 = plot(mu_arr, E_log_C_fixed_network_arr, '-.', 'color', colors(2, :), 'linewidth', 2);

    plot(x_A, y_A, '.', 'color', 'k', 'markersize', 22)
    text(x_A * 0.98, y_A * 1.004, '$$O$$', 'interpreter', 'latex', 'fontsize', 16)

    ax = gca;

    set(gca, 'xtick', [], 'fontsize', 12)
    set(gca, 'xticklabel', {''}, 'fontsize', 12)
    set(gca, 'ytick', sort([]), 'fontsize', 12)
    set(gca, 'yticklabel', {''}, 'fontsize', 12)

    xlim([min(mu_arr), max(mu_arr)])
    ylim([min(E_log_C_arr) * 0.99, max(E_log_C_arr)])

    ylabel('$$E[y]$$', 'interpreter', 'latex', 'fontsize', 20)
    xlabel('$$\mu_5$$', 'interpreter', 'latex', 'fontsize', 20)
    legend([h1 h2], {'Flexible network', 'Network fixed as at point $$O$$'}, 'interpreter', 'latex', '
        location', 'northwest', 'fontsize', 16)

    if save_flag == 1
        exportgraphics(gca, '../../../output_figures/fig3/E_log_C_mean.eps')
        exportgraphics(gca, '../../../output_figures/fig3/E_log_C_mean.png')
    end

    xx = [200, 400, 500, 400];
    close(figure(33))
    figure(33)

    colors = get(gca, 'ColorOrder');
    set(gcf, 'Position', xx)
    set(gca, 'TickLabelInterpreter', 'latex');
```

```
        box on
        grid on
        hold on

        x_A = mu_arr(1);
        y_A = V_log_C_arr(1);
        plot(mu_arr, y_A * ones(size(mu_arr)), 'color', 'k', 'linewidth', 0.5)
        plot(mu_arr, V_log_C_arr, 'color', colors(1, :), 'linewidth', 2)
        plot(mu_arr, V_log_C_fixed_network_arr, '-.', 'color', colors(2, :), 'linewidth', 2)

        plot(x_A, y_A, '.', 'color', 'k', 'markersize', 22)
        text(x_A * 0.98, y_A * 1.01, '$$O$$', 'interpreter', 'latex', 'fontsize', 16)

        ax = gca;

        set(gca, 'xtick', [], 'fontsize', 12)
        set(gca, 'xticklabel', {''}, 'fontsize', 12)
        set(gca, 'ytick', sort([]), 'fontsize', 12)
        set(gca, 'yticklabel', {''}, 'fontsize', 12)

        xlim([min(mu_arr), max(mu_arr)])
        ylim([min(V_log_C_arr) * 0.98, max(V_log_C_arr) * 1.04])

        ylabel('$$V[y]$$', 'interpreter', 'latex', 'fontsize', 20)
        xlabel('$$\mu_5$$', 'interpreter', 'latex', 'fontsize', 20)

        if save_flag == 1
            exportgraphics(gca, '../../../output_figures/fig3/V_log_C_mean.eps')
            exportgraphics(gca, '../../../output_figures/fig3/V_log_C_mean.png')
        end

        xx = [200, 400, 500, 400];
        close(figure(34))
        figure(34)

        colors = get(gca, 'ColorOrder');
        set(gcf, 'Position', xx)
        set(gca, 'TickLabelInterpreter', 'latex');
        box on
```

```
    grid on
    hold on

    x_A = mu_arr(1);
    y_A = welfare_arr(1);
    plot(mu_arr, y_A * ones(size(mu_arr)), 'color', 'k', 'linewidth', 0.5)
    plot(mu_arr, welfare_arr, 'color', colors(1, :), 'linewidth', 2)
    plot(mu_arr, welfare_fixed_network_arr, '-.', 'color', colors(2, :), 'linewidth', 2)

    plot(x_A, y_A, '.', 'color', 'k', 'markersize', 22)
    text(x_A * 0.98, y_A * 1.03, '$$O$$', 'interpreter', 'latex', 'fontsize', 16)

    ax = gca;

    set(gca, 'xtick', [], 'fontsize', 12)
    set(gca, 'xticklabel', {''}, 'fontsize', 12)
    set(gca, 'ytick', sort([]), 'fontsize', 12)
    set(gca, 'yticklabel', {''}, 'fontsize', 12)

    xlim([min(mu_arr), max(mu_arr)])

    ylabel('Welfare', 'interpreter', 'latex', 'fontsize', 20)
    xlabel('$$\mu_5$$', 'interpreter', 'latex', 'fontsize', 20)

    if save_flag == 1
        exportgraphics(gca, '../../../output_figures/fig3/welfare_mean.eps')
        exportgraphics(gca, '../../../output_figures/fig3/welfare_mean.png')
    end
end

if example_sigma_flag == 1
    xx = [200, 400, 500, 400];
    close(figure(32))
    figure(32)

    colors = get(gca, 'ColorOrder');
    set(gcf, 'Position', xx)
    set(gca, 'TickLabelInterpreter', 'latex');
    box on
```

```
    grid on
    hold on

    x_A = sigma_arr(1);
    y_A = E_log_C_arr(1);
    plot(sigma_arr, y_A * ones(size(sigma_arr)), 'color', 'k', 'linewidth', 0.5)
    h1 = plot(sigma_arr, E_log_C_arr, 'color', colors(1, :), 'linewidth', 2);
    h2 = plot(sigma_arr, E_log_C_fixed_network_arr, '-.', 'color', colors(2, :), 'linewidth', 2);

    plot(x_A, y_A, '.', 'color', 'k', 'markersize', 22)
    text(x_A * 1.004, y_A * 1.0009, '$$O$$', 'interpreter', 'latex', 'fontsize', 16)

    ax = gca;
    set(gca, 'xtick', [], 'fontsize', 12)
    set(gca, 'xticklabel', {''}, 'fontsize', 12)
    set(gca, 'ytick', sort([]), 'fontsize', 12)
    set(gca, 'yticklabel', {''}, 'fontsize', 12)

    xlim([min(sigma_arr), max(sigma_arr)])
    ylim([min(E_log_C_arr) * 0.995, max(E_log_C_arr) * 1.01])

    ylabel('$$E[y]$$', 'interpreter', 'latex', 'fontsize', 20)
    xlabel('$$\Sigma_{55}$$', 'interpreter', 'latex', 'fontsize', 20)
    legend([h1 h2], {'Flexible network', 'Network fixed as at point $$O$$'}, 'interpreter', 'latex', '
        location', 'northwest', 'fontsize', 16)

    if save_flag == 1
        exportgraphics(gca, '../../../output_figures/fig3/E_log_C_var.eps')
        exportgraphics(gca, '../../../output_figures/fig3/E_log_C_var.png')
    end

    xx = [200, 400, 500, 400];
    close(figure(33))
    figure(33)

    colors = get(gca, 'ColorOrder');
    set(gcf, 'Position', xx)
    set(gca, 'TickLabelInterpreter', 'latex');
    box on
```

```
        grid on
        hold on

        x_A = sigma_arr(1);
        y_A = V_log_C_arr(1);
        plot(sigma_arr, y_A * ones(size(sigma_arr)), 'color', 'k', 'linewidth', 0.5)
        rat = 0.15;
        plot(sigma_arr, V_log_C_arr, 'color', colors(1, :), 'linewidth', 2)
        plot(sigma_arr, V_log_C_fixed_network_arr, '-.', 'color', colors(2, :), 'linewidth', 2)

        plot(x_A, y_A, '.', 'color', 'k', 'markersize', 22)
        text(x_A * 1.005, y_A * 1.002, '$$O$$', 'interpreter', 'latex', 'fontsize', 16)

        ax = gca;

        set(gca, 'xtick', [], 'fontsize', 12)
        set(gca, 'xticklabel', {''}, 'fontsize', 12)
        set(gca, 'ytick', sort([]), 'fontsize', 12)
        set(gca, 'yticklabel', {''}, 'fontsize', 12)

        xlim([min(sigma_arr), max(sigma_arr)])
        ylim([min([V_log_C_arr; V_log_C_fixed_network_arr]) * 0.998, max([V_log_C_arr;
            V_log_C_fixed_network_arr]) * 1.0])

        ylabel('$$V[y]$$', 'interpreter', 'latex', 'fontsize', 20)
        xlabel('$$\Sigma_{55}$$', 'interpreter', 'latex', 'fontsize', 20)

        if save_flag == 1
            exportgraphics(gca, '../../../output_figures/fig3/V_log_C_var.eps')
            exportgraphics(gca, '../../../output_figures/fig3/V_log_C_var.png')
        end

        xx = [200, 400, 500, 400];
        close(figure(34))
        figure(34)

        colors = get(gca, 'ColorOrder');
        set(gcf, 'Position', xx)
        set(gca, 'TickLabelInterpreter', 'latex');
```

```
        box on
        grid on
        hold on

        x_A = sigma_arr(1);
        y_A = welfare_arr(1);
        plot(sigma_arr, y_A * ones(size(sigma_arr)), 'color', 'k', 'linewidth', 0.5)
        plot(sigma_arr, welfare_arr, 'color', colors(1, :), 'linewidth', 2)
        plot(sigma_arr, welfare_fixed_network_arr, '-.', 'color', colors(2, :), 'linewidth', 2)

        plot(x_A, y_A, '.', 'color', 'k', 'markersize', 22)
        text(x_A * 1.004, y_A * 1.0015, '$$O$$', 'interpreter', 'latex', 'fontsize', 16)

        ax = gca;

        set(gca, 'xtick', [], 'fontsize', 12)
        set(gca, 'xticklabel', {''}, 'fontsize', 12)
        set(gca, 'ytick', sort([]), 'fontsize', 12)
        set(gca, 'yticklabel', {''}, 'fontsize', 12)

        xlim([min(sigma_arr), max(sigma_arr)])
        ylim([min(welfare_arr) * 0.985, max(welfare_arr) * 1.004])

        ylabel('Welfare', 'interpreter', 'latex', 'fontsize', 20)
        xlabel('$$\Sigma_{55}$$', 'interpreter', 'latex', 'fontsize', 20)

        if save_flag == 1
            exportgraphics(gca, '../../../output_figures/fig3/welfare_var.eps')
            exportgraphics(gca, '../../../output_figures/fig3/welfare_var.png')
        end
    end
end
```

11.5.2 compute_eq.m

```
Input Variables
param: Structure containing all model parameters including n.
alpha_star_init: Initial guess for the input shares.
```

```

Output Variables
alpha_star: Equilibrium input shares for each firm after convergence.
conv_flag: Flag indicating whether the algorithm converged (1) or not (0).

Auxiliary Code
solve_firm_problem: Custom function called within the loop to solve the optimization problem for each
    firm, given the current input shares. This function computes the optimal input shares for each firm.
```

```matlab
function [alpha_star, conv_flag] = compute_eq(param, alpha_star_init)

% Compute the equilibrium
conv_flag = 1;

% Initialize expected benefit of each firm's input shares
% First index firm, second index input
alpha_star = alpha_star_init;

% Convergence flag and iteration parameters
has_converged = false;
iter = 0;
iter_max = 200;
tol = 1e-8;

% Iterate until convergence or maximum iterations reached
while has_converged == false && iter < iter_max
    iter = iter + 1;

    % Initialize new input shares matrix
    alpha_star_new = zeros(param.n, param.n);

    % Solve each firm's problem given the current input shares
    for i = 1:param.n
        alpha_star_new(i, :) = solve_firm_problem(param, i, alpha_star);
    end

    % Compute the maximum difference between old and new input shares
    max_diff = max(abs(alpha_star_new - alpha_star), [], 'all');
    if max_diff < tol
```

```
        has_converged = true;
    else
        alpha_star = alpha_star_new;

        % Update rule to help convergence
        alpha_star = 0.5 * alpha_star_new + 0.5 * alpha_star;
    end
end

% Check if maximum iterations were reached without convergence
if iter >= iter_max
    disp("No convergence")
    conv_flag = 0;
end

end
```

11.5.3 solve_firm_problem.m

```
Input Variables
param: Structure containing all model parameters including mu, sigma, n, rho, LS_min, H_inv, alpha_bar, and beta.
i_firm: Index of the firm for which the problem is being solved.
alpha_star: Current input shares for all firms.

Output Variables
alpha_chosen: Optimized input shares for the given firm.

Auxiliary Code
a_alpha_star: Custom function called to compute the equilibrium TFP of the firms given the current input shares and parameters.
quadprog: MATLAB built-in function used to solve the quadratic programming problem.
```

```
function [alpha_chosen] = solve_firm_problem(param, i_firm, alpha_star)
% Solve the problem of the firm using quadratic solver
% See app.lyx for notation (not the notation from model.lyx!! We need to harmonize those)

% The matrix in the quadratic optimization is often slightly (1e-15) not symmetric...
```

```matlab
warning('off', 'optim:quadprog:HessianNotSym')

% Extract parameters
mu = param.mu;
sigma = param.sigma;
n = param.n;
rho = param.rho;
LS_min = param.LS_min;
H_inv = param.H_inv;
alpha_bar = param.alpha_bar;
beta = param.beta;

% Compute the matrix associated with the TFP b
B_bar = -inv(H_inv(:, :, i_firm)) * alpha_bar(i_firm, :)';
A_bar = inv(H_inv(:, :, i_firm)) / 2;

% Compute the equilibrium TFP of the firms
a_star = a_alpha_star(alpha_star, param);

% Compute the Leontief inverse
L = inv(eye(n, n) - alpha_star);
one_i = zeros(n, 1);
one_i(i_firm) = 1;

% The linear part of the quadratic equation
f = -(B_bar + L * (mu + a_star - sigma * (one_i - L' * (one_i + (1 - rho) * beta))));

% Now the quadratic part. (There is a 2 since MATLAB expects 1/2 x'Hx)
H = 2 * (1/2 * L * sigma * L' - A_bar);

% Constraints for the optimization
A_const = ones(1, n);
b_const = (1 - LS_min);
options = optimoptions('quadprog', 'Display', 'off');

% Solve the quadratic programming problem
alpha_chosen = quadprog(H, f, A_const, b_const, [], [], zeros(n, 1), ones(n, 1), [], options);

end
```

11.5.4 a_alpha_star.m

```
Input Variables
alpha_star: Current input shares for all firms.
param: Structure containing all model parameters including n, H_inv, and alpha_bar.

Output Variables
a_star: Equilibrium TFP term for each firm.

Auxiliary Code
No additional functions are called; all operations are performed using built-in MATLAB commands and the provided parameters.
```

```
function [a_star] = a_alpha_star(alpha_star, param)
% Compute a(alpha_star), the equilibrium TFP term coming from the input choice

% Extract parameters
n = param.n;
H_inv = param.H_inv;
alpha_bar = param.alpha_bar;

% Initialize the equilibrium TFP term
a_star = zeros(n, 1);

% Compute the TFP term for each firm
for j = 1:n
    % Compute the temporary alpha vector for firm j
    alpha_temp = alpha_star(j, :)';
    % Compute the equilibrium TFP term for firm j
    a_star(j) = 0.5 * (alpha_temp - alpha_bar(j, :)')' * inv(H_inv(:, :, j)) * (alpha_temp - alpha_bar(j, :)');
end

end
```

11.5.5 compute_moments.m

```
Input Variables
alpha_star: Current input shares for all firms.
param: Structure containing all model parameters including mu, sigma, n, and beta.

Output Variables
E_log_C: Expected log consumption.
V_log_C: Variance of log consumption.
mean_LS: Mean labor share.
mean_log_P: Mean log prices.
covar_log_P: Covariance of log prices.

Auxiliary Code
a_alpha_star: Custom function called to compute the equilibrium TFP term given the current input shares
    and parameters.
```

```
function [E_log_C, V_log_C, mean_LS, mean_log_P, covar_log_P] = compute_moments(alpha_star, param)
% Compute various moments of the economy

% Extract parameters
mu = param.mu;
sigma = param.sigma;
n = param.n;
beta = param.beta;

% Compute the Leontief inverse
L = inv(eye(n, n) - alpha_star);

% Compute the equilibrium TFP term
a_star = a_alpha_star(alpha_star, param);

% Compute expected log consumption
E_log_C = beta' * L * (mu + a_star);

% Compute variance of log consumption
V_log_C = beta' * L * sigma * L' * beta;

% Compute mean labor share
```

```
mean_LS = (n - sum(alpha_star(:))) / n;

% Compute mean log prices
mean_log_P = -L * (mu + a_alpha_star(alpha_star, param));

% Compute covariance of log prices
covar_log_P = L * sigma * L';

end
```

11.6 Quantitative analysis

11.6.1 Replication.m

```
Input:
Files in various directories (Dir.Data, Dir.Input, Dir.Output, Dir.DataPr) containing sector data (e.g.,
    real gross output, real intermediate input, etc.)
Directory paths for different data types.
Source and variable types used in Getdata.m function to fetch data.

Output:
Files generated in the Processed Data folder containing harmonized sector data from 1948-2020.
Final message indicating whether the data processing was successful or not.

Auxiliary:
Functions like Getdata, Processdata, Mergedata, and Storedata are used extensively.
Directory structure setup ensuring the code works on any machine.
Loading data using specific functions and then processing it using custom logic defined in Processdata.m.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Date : September 2022
% Paper : Endogenous Production Networks under Supply Chain Uncertainty
%        Kopytov, Mishra, Nimark, and Taschereau-Dumouchel
% Replication code

%%
tic;

```

```
clc;
clear all;
close all;
format compact;

SectorAggregateData

TFP

clear;
close all;

toc;
```

11.6.2 TFP.m

```
Input:
'37 Sector Data' files from the processed data folder.
Various economic indicators such as real gross output, real intermediate inputs, employment, labor share,
     etc.

Output:
TFP (Total Factor Productivity) data stored in TFP_GO_nsm_nn.xlsx.
Messages indicating the success or failure of the data generation process.

Auxiliary:
Functions to fetch and process data (Getdata, Processdata).
Directory paths to locate input and output files.
Logic to calculate TFP using various economic indicators and processing steps.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bineet Mishra, September 2021
% Paper : Endogenous Production Networks under Uncertainty
%        Kopytov, Mishra, Nimark, and Taschereau-Dumouchel
% Input : 37 Sector Data
% Code : This generates the TFP process
% Output: TFP
% Note : This code mimics the STATA code (DATA_BUILD_bea_to_stata_37.do)
```

```
%       provided in replication folder by Lehn and Winberry (2020)

%% Good Practice
% tic;
% clc;
% clear;
% close all;
% format compact;

%% Keep the directory structure such that code works in any machine
Dir.Working = pwd;
Dir.Data   = '../../Processed Data';
Dir.Input = '../Input';
Dir.Output = '../Output/TFPMatlab';
Dir.DataPr = '../../Processed Data/TFPMatlab';

%% Total Factor Productivity
% Set the source and variable
Source           = 'KMNTData';
Var              = 'TFP';
% Get the data
[Table37SecAggData,Read37SecAggStatus] = Getdata(Dir,Var,Source); % ReadStatus: 1 successful, 0: Unsuccessful
% Process the data
Data37SecTFP     = Processdata(Table37SecAggData,Dir,Var);
% Store the data
Write37SecTFPStatus = Storedata(Data37SecTFP,Dir,Var); % WriteStatus: 1 successful, 0: Unsuccessful
% Display results
if  Write37SecTFPStatus == 1
   fprintf('TFP code run successful.\nProcessed Data-TFPMatlab folder must now have TFP_GO_nsm_nn.xlsx which contains harmonized TFP data for 37 sectors from 1948-2020.\n');
else
   fprintf('Check');
end

% toc;
```

11.6.3 Check.m

```
Input:
Generated TFP data files from both Matlab and Stata.
Other processed data files for real VA, nominal VA, real GO, etc.

Output:
Printed messages comparing the TFP data generated by Matlab and Stata.
Comparison results for the period 1948-2020 and 1948-2018 for various data sets like real VA, nominal VA, real GO, nominal GO, etc.

Auxiliary:
Functions to read and compare data from Excel files.
Logic to compute maximum differences between datasets to validate the consistency.
Directory paths to locate and store comparison results.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bineet Mishra, October 2021
% Paper  : Endogenous Production Networks under Uncertainty
%          Kopytov, Mishra, Nimark, and Taschereau-Dumouchel
% Code   : To check merged data are ok

tic;
clc;
clear;
close all;
format compact;

% BM: Keep the directory structure such that code works in any machine
Dir.Working = pwd;
Dir.Data   = '../../Raw Data';
Dir.Input  = '../Input';
Dir.Output = '../Output';
Dir.DataPr = '../../Processed Data';
Dir.TFPMat = '../../Processed Data/TFPMatlab';
Dir.TFPSta = '../../Processed Data/TFPStata';

%% Check the TFP data generated by Matlab and Stata
cd(Dir.TFPMat)
```

```
HarmonizedData_TFPMatlab = readtable('TFP_GO_nsm_nn.xlsx');

cd(Dir.Working);

cd(Dir.TFPSta)
HarmonizedData_TFPStata = readtable('TFP_GO_nsm_nn.xlsx');

cd(Dir.Working);
TFPMatlab             = HarmonizedData_TFPMatlab{1:end,2:end};
TFPStata              = HarmonizedData_TFPStata{1:end,2:end};
TFPdiffmaxsource      = max(max(TFPMatlab-TFPStata));
if TFPdiffmaxsource < 10^-3
   fprintf('Maximum difference is:');
   disp(TFPdiffmaxsource);
   fprintf('TFP generated by Matlab and Stata are identical\n');
else
   fprintf('Maximum difference is:');
   disp(TFPdiffmaxsource);
   fprintf('Check: TFP generated by Matlab and Stata are different\n');
end

%% Check the TFP data for sample period 1948-2020 and 1948-2018
cd(Dir.TFPSta)
HarmonizedData_TFPStata4820 = readtable('TFP_GO_nsm_nn.xlsx');

cd(Dir.Working);

Sourcefile = strcat(Dir.Data,'/TFP_GO_nsm_nn_4818.xls');
Destination = Dir.Input;
copyfile(Sourcefile,Destination);
cd(Dir.Input)
HarmonizedData_TFPStata4818 = readtable('TFP_GO_nsm_nn_4818.xls');

cd(Dir.Working);
TFPStata4820             = HarmonizedData_TFPStata{1:end-2,2:end};
TFPStata4818             = HarmonizedData_TFPStata4818{1:end,2:end};
TFPdiffmaxsample         = max(max(TFPStata4820-TFPStata4818));
if TFPdiffmaxsample < 10^-1
   fprintf('Maximum difference is:');
```

```
    disp(TFPdiffmaxsample);
    fprintf('TFP for 1948-2020 and 1948-2018 are almost identical\n');
else
    fprintf('Maximum difference is:');
    disp(TFPdiffmaxsample);
    fprintf('Check: TFP for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Real VA data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_RealVA = readtable('37 Sector Data.xlsx','Sheet','real_va');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_RealVA = readtable('37 Sector Data vLW.xlsx','Sheet','real_va');

cd(Dir.Working);
RealVA4820            = HarmonizedData4820_RealVA{1:end-2,2:end};
RealVA4818            = HarmonizedData4818_RealVA{1:end,2:end};
RealVAdiffmaxsample   = max(max(RealVA4820-RealVA4818));
if RealVAdiffmaxsample < 50
    fprintf('Maximum difference is:');
    disp(RealVAdiffmaxsample);
    fprintf('Real VA for 1948-2020 and 1948-2018 are almost identical\n');
else
    fprintf('Maximum difference is:');
    disp(RealVAdiffmaxsample);
    fprintf('Check: Real VA for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Nominal VA data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_NominalVA = readtable('37 Sector Data.xlsx','Sheet','nominal_va');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_NominalVA = readtable('37 Sector Data vLW.xlsx','Sheet','nominal_va');
```

```
cd(Dir.Working);
NominalVA4820            = HarmonizedData4820_NominalVA{1:end-2,2:end};
NominalVA4818            = HarmonizedData4818_NominalVA{1:end,2:end};
NominalVAdiffmaxsample   = max(max(NominalVA4820-NominalVA4818));
if NominalVAdiffmaxsample < 10^5
   fprintf('Maximum difference is:');
   disp(NominalVAdiffmaxsample);
   fprintf('Nominal VA for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(NominalVAdiffmaxsample);
   fprintf('Check: Nominal VA for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Real GO data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_RealGO = readtable('37 Sector Data.xlsx','Sheet','real_GO');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_RealGO = readtable('37 Sector Data vLW.xlsx','Sheet','real_GO');

cd(Dir.Working);
RealGO4820            = HarmonizedData4820_RealGO{1:end-2,2:end};
RealGO4818            = HarmonizedData4818_RealGO{1:end,2:end};
RealGOdiffmaxsample   = max(max(RealGO4820-RealGO4818));
if RealGOdiffmaxsample < 50
   fprintf('Maximum difference is:');
   disp(RealGOdiffmaxsample);
   fprintf('Real GO for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(RealGOdiffmaxsample);
   fprintf('Check: Real GO for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Nominal GO data for sample period 1948-2020 and 1948-2018
```

```
cd(Dir.DataPr)
HarmonizedData4820_NominalGO = readtable('37 Sector Data.xlsx','Sheet','nominal_GO');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_NominalGO = readtable('37 Sector Data vLW.xlsx','Sheet','nominal_GO');

cd(Dir.Working);
NominalGO4820            = HarmonizedData4820_NominalGO{1:end-2,2:end};
NominalGO4818            = HarmonizedData4818_NominalGO{1:end,2:end};
NominalGOdiffmaxsample   = max(max(NominalGO4820-NominalGO4818));
if NominalGOdiffmaxsample < 50
    fprintf('Maximum difference is:');
    disp(NominalGOdiffmaxsample);
    fprintf('Nominal GO for 1948-2020 and 1948-2018 are almost identical\n');
else
    fprintf('Maximum difference is:');
    disp(NominalGOdiffmaxsample);
    fprintf('Check: Nominal GO for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Real II data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_RealII = readtable('37 Sector Data.xlsx','Sheet','real_II');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_RealII = readtable('37 Sector Data vLW.xlsx','Sheet','real_II');

cd(Dir.Working);
RealII4820            = HarmonizedData4820_RealII{1:end-2,2:end};
RealII4818            = HarmonizedData4818_RealII{1:end,2:end};
RealIIdiffmaxsample    = max(max(RealII4820-RealII4818));
if RealGOdiffmaxsample < 50
    fprintf('Maximum difference is:');
    disp(RealIIdiffmaxsample);
    fprintf('Real II for 1948-2020 and 1948-2018 are almost identical\n');
```

```
else
    fprintf('Maximum difference is:');
    disp(RealIIdiffmaxsample);
    fprintf('Check: Real II for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Nominal VA data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_NominalII = readtable('37 Sector Data.xlsx','Sheet','nominal_II');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_NominalII = readtable('37 Sector Data vLW.xlsx','Sheet','nominal_II');

cd(Dir.Working);
NominalII4820            = HarmonizedData4820_NominalII{1:end-2,2:end};
NominalII4818            = HarmonizedData4818_NominalII{1:end,2:end};
NominalIIdiffmaxsample   = max(max(NominalII4820-NominalII4818));
if NominalIIdiffmaxsample < 50
    fprintf('Maximum difference is:');
    disp(NominalIIdiffmaxsample);
    fprintf('Nominal II for 1948-2020 and 1948-2018 are almost identical\n');
else
    fprintf('Maximum difference is:');
    disp(NominalIIdiffmaxsample);
    fprintf('Check: Nominal II for 1948-2020 and 1948-2018 are different\n');
end

%% Check the II shares data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_IIShares = readtable('37 Sector Data.xlsx','Sheet','II_shares');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_IIShares = readtable('37 Sector Data vLW.xlsx','Sheet','II_shares');

cd(Dir.Working);
```

```
IIShares4820            = HarmonizedData4820_IIShares{1:end-2,2:end};
IIShares4818            = HarmonizedData4818_IIShares{1:end,2:end};
IISharesdiffmaxsample   = max(max(IIShares4820-IIShares4818));
if IISharesdiffmaxsample < 10^-1
   fprintf('Maximum difference is:');
   disp(IISharesdiffmaxsample);
   fprintf('II Shares for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(IISharesdiffmaxsample);
   fprintf('Check: II Shares for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Employment data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_employment = readtable('37 Sector Data.xlsx','Sheet','employment');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_employment = readtable('37 Sector Data vLW.xlsx','Sheet','employment');

cd(Dir.Working);
Employment4820            = HarmonizedData4820_employment{1:end-2,2:end};
Employment4818            = HarmonizedData4818_employment{1:end,2:end};
Employmentdiffmaxsample   = max(max(Employment4820-Employment4818));
if Employmentdiffmaxsample < 50
   fprintf('Maximum difference is:');
   disp(Employmentdiffmaxsample);
   fprintf('Employment for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(Employmentdiffmaxsample);
   fprintf('Check: Employment for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Labor Share Unscaled data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_LSU = readtable('37 Sector Data.xlsx','Sheet','labor_share_unscaled');
```

```
cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_LSU = readtable('37 Sector Data vLW.xlsx','Sheet','labor_share_unscaled');

cd(Dir.Working);
LSU4820             = HarmonizedData4820_LSU{1:end-2,2:end};
LSU4818             = HarmonizedData4818_LSU{1:end,2:end};
LSUdiffmaxsample    = max(max(LSU4820-LSU4818));
if LSUdiffmaxsample < 10^-1
   fprintf('Maximum difference is:');
   disp(LSUdiffmaxsample);
   fprintf('Labor Share Unscaled for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(LSUdiffmaxsample);
   fprintf('Check: Labor Share Unscaled for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Scaling factor data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_SF = readtable('37 Sector Data.xlsx','Sheet','scalingfactor');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_SF = readtable('37 Sector Data vLW.xlsx','Sheet','scaling_factor');

cd(Dir.Working);
SF4820             = HarmonizedData4820_SF{1,1:end};
SF4818             = HarmonizedData4818_SF{1,2:end};
SFdiffmaxsample    = max(max(SF4820-SF4818));
if SFdiffmaxsample < 10^-1
   fprintf('Maximum difference is:');
   disp(SFdiffmaxsample);
   fprintf('Scaling factor for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
```

```
    disp(LSUdifSFdiffmaxsamplefmaxsample);
    fprintf('Check: Scaling factor for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Labor Share data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_LS = readtable('37 Sector Data.xlsx','Sheet','labor_share');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_LS = readtable('37 Sector Data vLW.xlsx','Sheet','labor_share');

cd(Dir.Working);
LS4820             = HarmonizedData4820_LS{1:end-2,2:end};
LS4818             = HarmonizedData4818_LS{1:end,2:end};
LSdiffmaxsample    = max(max(LS4820-LS4818));
if LSdiffmaxsample < 10^-1
    fprintf('Maximum difference is:');
    disp(LSdiffmaxsample);
    fprintf('Labor Share for 1948-2020 and 1948-2018 are almost identical\n');
else
    fprintf('Maximum difference is:');
    disp(LSdiffmaxsample);
    fprintf('Check: Labor Share for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Nominal Capital data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_NominalCapital = readtable('37 Sector Data.xlsx','Sheet','nominal_capital');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_NominalCapital = readtable('37 Sector Data vLW.xlsx','Sheet','nominal_capital');

cd(Dir.Working);
NominalCapital4820           = HarmonizedData4820_NominalCapital{1:end-2,2:end};
NominalCapital4818           = HarmonizedData4818_NominalCapital{1:end,2:end};
```

```
NominalCapitaldiffmaxsample  = max(max(NominalCapital4820-NominalCapital4818));
if NominalCapitaldiffmaxsample < 50
   fprintf('Maximum difference is:');
   disp(NominalCapitaldiffmaxsample);
   fprintf('Nominal Capital for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(NominalCapitaldiffmaxsample);
   fprintf('Check: Nominal Capital for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Depreciation Rate data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_depreciationrates = readtable('37 Sector Data.xlsx','Sheet','depreciation_rates');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_depreciationrates = readtable('37 Sector Data vLW.xlsx','Sheet','depreciation_rates');

cd(Dir.Working);
Depreciationrates4820          = HarmonizedData4820_depreciationrates{1:end-2,2:end};
Depreciationrates4818          = HarmonizedData4818_depreciationrates{1:end-7,2:end};
Depreciationratesdiffmaxsample  = max(max(Depreciationrates4820-Depreciationrates4818));
if Depreciationratesdiffmaxsample < 10^-1
   fprintf('Maximum difference is:');
   disp(Depreciationratesdiffmaxsample);
   fprintf('Depreciation Rate for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(Depreciationratesdiffmaxsample);
   fprintf('Check: Depreciation Rate for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Nominal Investment data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_NominalInvestment = readtable('37 Sector Data.xlsx','Sheet','nominal_inv');

cd(Dir.Working);
```

```
cd(Dir.Input)
HarmonizedData4818_NominalInvestment = readtable('37 Sector Data vLW.xlsx','Sheet','nominal_inv');

cd(Dir.Working);
NominalInvestment4820           = HarmonizedData4820_NominalInvestment{1:end-2,2:end};
NominalInvestment4818           = HarmonizedData4818_NominalInvestment{1:end-76,2:end-40};
NominalInvestmentdiffmaxsample  = max(max(NominalInvestment4820-NominalInvestment4818));
if NominalInvestmentdiffmaxsample < 50
   fprintf('Maximum difference is:');
   disp(NominalInvestmentdiffmaxsample);
   fprintf('Nominal Investment for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(NominalInvestmentdiffmaxsample);
   fprintf('Check: Nominal Investment for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Real Investment data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_RealInvestment = readtable('37 Sector Data.xlsx','Sheet','real_inv');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_RealInvestment = readtable('37 Sector Data vLW.xlsx','Sheet','real_inv');

cd(Dir.Working);
RealInvestment4820           = HarmonizedData4820_RealInvestment{1:end-2,2:end};
RealInvestment4818           = HarmonizedData4818_RealInvestment{1:end,2:end};
RealInvestmentdiffmaxsample  = max(max(RealInvestment4820-RealInvestment4818));
if RealInvestmentdiffmaxsample < 50
   fprintf('Maximum difference is:');
   disp(RealInvestmentdiffmaxsample);
   fprintf('Real Investment for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(RealInvestmentdiffmaxsample);
   fprintf('Check: Real Investment for 1948-2020 and 1948-2018 are different\n');
```

```
end

%% Check the Real Investment data for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_RealInvestmentDollars = readtable('37 Sector Data.xlsx','Sheet','real_inv_dollars');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_RealInvestmentDollars = readtable('37 Sector Data vLW.xlsx','Sheet','real_inv_dollars'
    );

cd(Dir.Working);
RealInvestmentDollars4820          = HarmonizedData4820_RealInvestmentDollars{1:end-2,2:end};
RealInvestmentDollars4818          = HarmonizedData4818_RealInvestmentDollars{1:end,2:end};
RealInvestmentDollarsdiffmaxsample = max(max(RealInvestmentDollars4820-RealInvestmentDollars4818));
if RealInvestmentDollarsdiffmaxsample < 50
   fprintf('Maximum difference is:');
   disp(RealInvestmentDollarsdiffmaxsample);
   fprintf('Real Investment Dollars for 1948-2020 and 1948-2018 are almost identical\n');
else
   fprintf('Maximum difference is:');
   disp(RealInvestmentDollarsdiffmaxsample);
   fprintf('Check: Real Investment Dollars for 1948-2020 and 1948-2018 are different\n');
end

%% Check the Price VA for sample period 1948-2020 and 1948-2018
cd(Dir.DataPr)
HarmonizedData4820_PriceVA = readtable('37 Sector Data.xlsx','Sheet','VA_P');

cd(Dir.Working);

cd(Dir.Input)
HarmonizedData4818_PriceVA = readtable('37 Sector Data vLW.xlsx','Sheet','VA_P');

cd(Dir.Working);
PriceVA4820            = HarmonizedData4820_PriceVA{1:end-2,2:end};
PriceVA4818            = HarmonizedData4818_PriceVA{1:end,2:end};
PriceVAdiffmaxsample   = max(max(PriceVA4820-PriceVA4818));
```

```
if RealInvestmentDollarsdiffmaxsample < 10
    fprintf('Maximum difference is:');
    disp(PriceVAdiffmaxsample);
    fprintf('Price VA for 1948-2020 and 1948-2018 are almost identical\n');
else
    fprintf('Maximum difference is:');
    disp(PriceVAdiffmaxsample);
    fprintf('Check: Price VA for 1948-2020 and 1948-2018 are different\n');
end

%%
cd(Dir.Data);
Cons37Sec4818 = load('IOmat4718dat_37sec.mat','Cons47bea');
cd(Dir.Working);

cd(Dir.DataPr);
Cons37Sec4820 = load('IOmat4720dat_37sec.mat','Cons47bea');
Alpha4818   = load('IOmat4720dat_37sec.mat','ALPHA');
Alpha4820   = load('IOmat4720dat_37sec.mat','ALPHA4720');
cd(Dir.Working);


%%
T = 1947:1:2020;
figure
for s = 1:37
    subplot(10,4,s)
    plot(T,Cons37Sec4820.Cons47bea(:,s));
end
figure
for s = 1:37
    subplot(10,4,s)
    plot(T(1:end-2),Cons37Sec4818.Cons47bea(:,s));
end

%%
[vLWalpharidx,vLWalphacidx,vLWalphahidx] = find(Alpha4818.ALPHA<0);
[KMNTalpharidx,KMNTalphacidx,KMNTalphahidx] = ind2sub(size(Alpha4820.ALPHA4720),find(Alpha4820.ALPHA4720
    <0));
```

```
clear;
close all;
toc;
```

11.6.4 Getdata.m

```
Input:
Directory paths for different data types.
Source and variable types to fetch the corresponding data.

Output:
Data tables for various economic indicators fetched from specified directories.

Auxiliary:
Logic to handle different sources and variable types.
Functions to copy and read data from Excel files based on specified criteria.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bineet Mishra, September 2021
% Paper  : Endogenous Production Networks under Uncertainty
%          Kopytov, Mishra, Nimark, and Taschereau-Dumouchel
% Function: a) To copy data from Raw Data folder or Processed Data to Input Folder
%           b) To read the data from the files

function [Data,Status] = Getdata(Dir,Var,Source)
    switch Source
        case 'Recent'
            switch Var
                case 'RealVA'
                    Sourcefile = strcat(Dir.Data,'/Chain Quantity Indexes VA by Ind 1997-2020.xlsx');
                    Destination = Dir.Input;
                    Status     = copyfile(Sourcefile,Destination);
                    cd(Dir.Input);
                    Data       = readtable('Chain Quantity Indexes VA by Ind 1997-2020.xlsx','
                        ReadVariableNames',false);
                case 'RealGO'
                    Sourcefile = strcat(Dir.Data,'/Chain Quantity Indexes GO by Ind 1997-2020.xlsx');
```

```
                Destination = Dir.Input;
                Status     = copyfile(Sourcefile,Destination);
                cd(Dir.Input);
                Data       = readtable('Chain Quantity Indexes GO by Ind 1997-2020.xlsx','ReadVariableNames',false);
            case 'RealII'
                Sourcefile = strcat(Dir.Data,'/Chain Quantity Indexes II by Ind 1997-2020.xlsx');
                Destination = Dir.Input;
                Status     = copyfile(Sourcefile,Destination);
                cd(Dir.Input);
                Data       = readtable('Chain Quantity Indexes II by Ind 1997-2020.xlsx','ReadVariableNames',false);
            case 'NominalVA'
                Sourcefile = strcat(Dir.Data,'/Nom VA by Ind 1997-2020.xlsx');
                Destination = Dir.Input;
                Status     = copyfile(Sourcefile,Destination);
                cd(Dir.Input);
                Data       = readtable('Nom VA by Ind 1997-2020.xlsx','ReadVariableNames',false);
            case 'NominalGO'
                Sourcefile = strcat(Dir.Data,'/Nom GO by Ind 1997-2020.xlsx');
                Destination = Dir.Input;
                Status     = copyfile(Sourcefile,Destination);
                cd(Dir.Input);
                Data       = readtable('Nom GO by Ind 1997-2020.xlsx','ReadVariableNames',false);
            case 'NominalII'
                Sourcefile = strcat(Dir.Data,'/Nom II by Ind 1997-2020.xlsx');
                Destination = Dir.Input;
                Status     = copyfile(Sourcefile,Destination);
                cd(Dir.Input);
                Data       = readtable('Nom II by Ind 1997-2020.xlsx','ReadVariableNames',false);
            case 'Employment'
                Sourcefile = strcat(Dir.Data,'/FTPT by Ind 1998-2020.xlsx');
                Destination = Dir.Input;
                Status = copyfile(Sourcefile,Destination);
                cd(Dir.Input);
                Data   = readtable('FTPT by Ind 1998-2020.xlsx','ReadVariableNames',false);
            case 'LaborSharesUnScaled'
                Sourcefile = strcat(Dir.Data,'/VA Components by Ind 1997-2020.xlsx');
                Destination = Dir.Input;
```

```
            Status     = copyfile(Sourcefile,Destination);
            cd(Dir.Input);
            Data       = readtable('VA Components by Ind 1997-2020.xlsx','ReadVariableNames',false);
        case 'SelfEmployed'
            Sourcefile = strcat(Dir.Data,'/Self Employed by Industry 1998-2020.xlsx');
            Destination = Dir.Input;
            Status9820 = copyfile(Sourcefile,Destination);
            cd(Dir.Input);
            Data9820   = readtable('Self Employed by Industry 1998-2020.xlsx','ReadVariableNames',
                false);
            Sourcefile = strcat(Dir.Data,'/Self Employed by Industry 1987-1997.xlsx');
            Destination = Dir.Input;
            Status8797 = copyfile(Sourcefile,Destination);
            cd(Dir.Input);
            Data8797   = readtable('Self Employed by Industry 1987-1997.xlsx','ReadVariableNames',
                false);
            Data.SE9820 = Data9820;
            Data.SE8797 = Data8797;
            Status.SE1 = Status9820;
            Status.SE2 = Status8797;
        case 'NominalCapital'
            Sourcefile = strcat(Dir.Data,'/Nominal Year End Net Stock Capital by Industry 1947-2020.
                xlsx');
            Destination = Dir.Input;
            Status     = copyfile(Sourcefile,Destination);
            cd(Dir.Input);
            Data       = readtable('Nominal Year End Net Stock Capital by Industry 1947-2020.xlsx','
                ReadVariableNames',false);
        case 'DepreciationRate'
            cd(Dir.Input);
            Data       = readtable('DepreciationRate.xlsx','ReadVariableNames',true);
        case 'NominalInvestment'
            cd(Dir.Input);
            Data       = readtable('NominalInvestment.xlsx','ReadVariableNames',true);
        case 'RealInvestment'
            Sourcefile = strcat(Dir.Data,'/Chain-Type Quantity Indexes for Investment in Private
                Fixed Assets by Industry 1947-2020.xlsx');
            Destination = Dir.Input;
            Status     = copyfile(Sourcefile,Destination);
```

```
                  cd(Dir.Input);
                  Data      = readtable('Chain-Type Quantity Indexes for Investment in Private Fixed Assets by Industry 1947-2020.xlsx','ReadVariableNames',false);
              otherwise
                  fprintf('Incorrect variable. Set Var from any one of the following: \n RealVA \n RealGO \n RealII \n NominalVA \n NominalGO \n NominalII \n Employment \n LaborSharesUnScaled \n SelfEmployed \n NominalCapital \n DepreciationRate \n NominalInvestment \n RealCapital \n');
           end
        case 'vLWData'
          Sourcefile = strcat(Dir.Data,'/37 Sector Data vLW.xlsx');
          Destination = Dir.Input;
          Status    = copyfile(Sourcefile,Destination);
          cd(Dir.Input);
           switch Var
              case 'RealVA'
                  Data      = readtable('37 Sector Data vLW.xlsx','Sheet','real_VA');
              case 'RealGO'
                  Data      = readtable('37 Sector Data vLW.xlsx','Sheet','real_GO');
              case 'RealII'
                  Data      = readtable('37 Sector Data vLW.xlsx','Sheet','real_II');
              case 'NominalVA'
                  Data      = readtable('37 Sector Data vLW.xlsx','Sheet','nominal_va');
              case 'NominalGO'
                  Data      = readtable('37 Sector Data vLW.xlsx','Sheet','nominal_GO');
              case 'NominalII'
                  Data      = readtable('37 Sector Data vLW.xlsx','Sheet','nominal_II');
              case 'Employment'
                  Data      = readtable('37 Sector Data vLW.xlsx','Sheet','employment');
              case 'LaborSharesUnScaled'
                  Data      = readtable('37 Sector Data vLW.xlsx','Sheet','labor_share_unscaled');
              case 'NominalInvestment'
                  Sourcefile = strcat(Dir.Data,'/Nominal Investment in Private Fixed Assets by Industry 1947-2020.xlsx');
                  Destination = Dir.Input;
                  Status    = copyfile(Sourcefile,Destination);
                  cd(Dir.Input);
                  Data = readtable('Nominal Investment in Private Fixed Assets by Industry 1947-2020.xlsx','ReadVariableNames',false);
```

```
            otherwise
                fprintf('Incorrect variable. Set Var from any one of the following: \n RealVA \n RealGO
                     \n RealII \n NominalVA \n NominalGO \n NominalII \n Employment \n
                     LaborSharesUnScaled \n SelfEmployed \n NominalCapital \n DepreciationRate \n
                     NominalInvestment \n RealCapital \n');
        end
    case 'KMNTData'
        Sourcefile = strcat(Dir.Data,'/37 Sector Data.xlsx');
        Destination = Dir.Input;
        Status     = copyfile(Sourcefile,Destination);
        cd(Dir.Input);

        real_GO         = readtable('37 Sector Data.xlsx','Sheet','real_GO');
        real_II         = readtable('37 Sector Data.xlsx','Sheet','real_II');
        II_shares       = readtable('37 Sector Data.xlsx','Sheet','II_shares');
        employment      = readtable('37 Sector Data.xlsx','Sheet','employment');
        labor_share     = readtable('37 Sector Data.xlsx','Sheet','labor_share');
        nominal_capital = readtable('37 Sector Data.xlsx','Sheet','nominal_capital');
        depreciation_rates = readtable('37 Sector Data.xlsx','Sheet','depreciation_rates');
        real_inv_dollars = readtable('37 Sector Data.xlsx','Sheet','real_inv_dollars');

        Data.real_GO         = real_GO;
        Data.real_II         = real_II;
        Data.II_shares       = II_shares;
        Data.employment      = employment;
        Data.labor_share     = labor_share;
        Data.nominal_capital = nominal_capital;
        Data.depreciation_rates = depreciation_rates;
        Data.real_inv_dollars = real_inv_dollars;
    otherwise
        fprintf('Incorrect variable. Set Source as Recent or vLWData or KMNTData\n');
    end
cd(Dir.Working);
end
```

11.6.5 LoadIOMat.m

```
Input:
```

```
Files from raw data directory containing Input-Output Accounts and Fixed Assets data.

Output:
Various IO matrices, value-added data, consumption data, investment data, and capital stock data.

Auxiliary:
Directory structure setup ensuring the code works on any machine.
Logic to load data from Excel files into Matlab.
Processing steps to create consistent IO tables and depreciation rates.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bineet Mishra, September 2021
% Paper : Endogenous Production Networks under Uncertainty
%        Kopytov, Mishra, Nimark, and Taschereau-Dumouchel
% Note : This code is borrowed from
% Christian vom Lehn, July 2020
% This m-file loads into Matlab the Use tables from the Input-Output
% Accounts at the BEA and Fixed Assets data. These are used to construct
% moments corresponding to production parameters in the framework of vom
% Lehn and Winberry (2020).
% Bineet: Changes are done a) to have data from 1947-2020
%                       b) work in any machine

%% Good Practice
% tic;
% clc;
% clear;
% close all;
% format compact

% BM: Keep the directory structure such that code works in any machine
Dir.Working = pwd;
Dir.Data  = '../../Raw Data';
Dir.Input = '../Input';
Dir.Output = '../Output';
Dir.DataPr = '../../Processed Data';

% BM: Copy these files from Raw Data folder to Input folder
Sourcefile = strcat(Dir.Data,'/IOUse_Before_Redefinitions_PRO_1947-1962_Summary.xlsx');
```

```
Destination = Dir.Input;
copyfile(Sourcefile,Destination);

Sourcefile = strcat(Dir.Data,'/IOUse_Before_Redefinitions_PRO_1963-1996_Summary.xlsx');
Destination = Dir.Input;
copyfile(Sourcefile,Destination);

%Sourcefile = strcat(Dir.Data,'/IOUse_Before_Redefinitions_PRO_1997-2018.xlsx');
Sourcefile = strcat(Dir.Data,'/IOUse_Before_Redefinitions_PRO_1997-2020_Summary.xlsx');
Destination = Dir.Input;
copyfile(Sourcefile,Destination);

Sourcefile = strcat(Dir.Data,'/DetailNonres_rate.xlsx');
Destination = Dir.Input;
copyfile(Sourcefile,Destination);

Sourcefile = strcat(Dir.Data,'/detailnonres_inv1.xlsx');
Destination = Dir.Input;
copyfile(Sourcefile,Destination);

Sourcefile = strcat(Dir.Data,'/detailnonres_inv2.xlsx');
Destination = Dir.Input;
copyfile(Sourcefile,Destination);

Sourcefile = strcat(Dir.Data,'/Nominal Year End Net Stock Capital by Industry 1947-2020.xlsx');
Destination = Dir.Input;
copyfile(Sourcefile,Destination);

Sourcefile = strcat(Dir.Data,'/detailnonres_stk1.xlsx');
Destination = Dir.Input;
copyfile(Sourcefile,Destination);

Sourcefile = strcat(Dir.Data,'/Constructing Residential Assets Depreciation Rate.xlsx');
Destination = Dir.Input;
copyfile(Sourcefile,Destination);

% BM: Change to input data path
cd(Dir.Input)

```

```
%%%Load in Excel data on Input Output Matrices

for i=3:18
   IOmat4762(:,:,i-2)=xlsread('IOUse_Before_Redefinitions_PRO_1947-1962_Summary',i,'C8:BL58');
end

for i=3:36
   IOmat6396(:,:,i-2)=xlsread('IOUse_Before_Redefinitions_PRO_1963-1996_Summary',i,'C8:CH77');
end

% BM: Data till 2020
%for i=1:22
for i=1:24
   %IOmat9718(:,:,i)=xlsread('IOUse_Before_Redefinitions_PRO_1997-2018',i,'C8:CV90');
   %BM: BEA has changed the data structure
   IOmat9720(:,:,i)=xlsread('IOUse_Before_Redefinitions_PRO_1997-2020_Summary',i,'C8:CR86');
end

% BM: Change to working path
cd(Dir.Working)

%%%Convert NaN observations (....) in the Excel files to zero
IOmat4762(isnan(IOmat4762))=0;
IOmat6396(isnan(IOmat6396))=0;
%IOmat9718(isnan(IOmat9718))=0;
IOmat9720(isnan(IOmat9720))=0;

%%%Create consistent IO tables for a series of classifications
% IO mat provides the Use matrix flows in current dollars (producers are
% row, purchasers are columns); Value Added provides the value added by
% industry (Gross Output similarly); Consumption provides the total final
% use of each industry's production for private consumption and exports;
% Structures (InvS) provides the total final use of each industry's production for
% structures (residential and non-residential); Equipment (InvE) and
% Intellectual Property (InvO) provide final use investment for each of
% these categories. NOTE: This does not include government final uses,
% imports, or inventories.

%%%1947-2018 IO Mat, BEA Data based (37 non-govt, non-farm sectors)
```

```
secnum = 37;

% BM: add yearnum and secname
startyear = 1947;
lastyear  = 2020;
yearnum   = lastyear-startyear+1;     % Number of years
years     = (startyear:1:lastyear)'; % Years
%  Same sector name as in 37 Sector Data.xlsx
secname   = {'Years','Mining','Utilities','Construction', 'WoodProducts',...
             'NonmetallicMinerals','PrimaryMetals','FabricatedMetals',...
             'Machinery','ComputerandElectronic', 'ElectricalEquipment',...
             'MotorVehicles','OtherTransequip','Furnitureandrelated',...
             'MiscMfg','Foodandbeverage','Textile','Apparel','Paper',...
             'Printing','Petroleum','Chemical','Plastics','WT','RT','TW',...
             'Info','FI','RE','ProfBus','Mgmt','Admin','Edu','Health',...
             'Arts','Accomm','FoodServ','Other'};

% IOmat47bea=zeros(secnum,secnum,72);
% VA47bea=zeros(72,secnum);
% GO47bea=zeros(72,secnum);
% Cons47bea=zeros(72,secnum);
% InvS47bea=zeros(72,secnum);
% InvE47bea=zeros(72,secnum);
% InvO47bea=zeros(72,secnum);
% BM: vLW's code has number of years hardcoded to 72
IOmat47bea=zeros(secnum,secnum,yearnum);
VA47bea=zeros(yearnum,secnum);
GO47bea=zeros(yearnum,secnum);
Cons47bea=zeros(yearnum,secnum);
InvS47bea=zeros(yearnum,secnum);
InvE47bea=zeros(yearnum,secnum);
InvO47bea=zeros(yearnum,secnum);

indcell = cell(secnum);
indcell{1}=(3:5);
for i=2:27
   indcell{i}=i+4;
end
```

```
indcell{28}=(32:33);
for i=29:secnum
   indcell{i}=i+5;
end

for i=1:16 %number of years in first Excel file
   for j=1:secnum
      for k=1:secnum
         IOmat47bea(j,k,i)=sum(sum(IOmat4762(indcell{j},indcell{k},i)));
      end
      VA47bea(i,j)=sum(IOmat4762(50,indcell{j},i)); % Row: Total Value Added
      GO47bea(i,j)=sum(IOmat4762(51,indcell{j},i)); % Row: Total industry Output
      Cons47bea(i,j)=sum(sum(IOmat4762(indcell{j},[48 53],i))); % Column: Personal consumption
          expenditures,Private fixed investment in structures,Private fixed investment in equipment,
          Private fixed investment in intellectual property products, Change in private inventories,
          Exports of goods and services
      InvS47bea(i,j)=sum(IOmat4762(indcell{j},49,i)); % Column: Private fixed investment in structures
      InvE47bea(i,j)=sum(IOmat4762(indcell{j},50,i)); % Column: Private fixed investment in equipment
      InvO47bea(i,j)=sum(IOmat4762(indcell{j},51,i)); % Column: Private fixed investment in intellectual
          property products
   end
end

indcell{25}=(29:36);
indcell{26}=(37:40);
indcell{27}=(41:44);
indcell{28}=(45:46);
indcell{29}=(47:49);
indcell{30}=50;
indcell{31}=(51:52);
indcell{32}=53;
indcell{33}=(54:56);
indcell{34}=(57:58);
indcell{35}=59;
indcell{36}=60;
indcell{37}=61;

for i=17:50
   for j=1:secnum
```

```
        for k=1:secnum
            IOmat47bea(j,k,i)=sum(sum(IOmat6396(indcell{j},indcell{k},i-16)));
        end
        VA47bea(i,j)=sum(IOmat6396(69,indcell{j},i-16)); % Row: Total Value Added
        GO47bea(i,j)=sum(IOmat6396(70,indcell{j},i-16)); % Row: Total industry Output
        Cons47bea(i,j)=sum(sum(IOmat6396(indcell{j},[67 72],i-16))); % Column: Personal consumption
             expenditures,Private fixed investment in structures,Private fixed investment in equipment,
             Private fixed investment in intellectual property products, Change in private inventories,
             Exports of goods and services
        InvS47bea(i,j)=sum(IOmat6396(indcell{j},68,i-16)); % Column: Private fixed investment in structures
        InvE47bea(i,j)=sum(IOmat6396(indcell{j},69,i-16)); % Column: Private fixed investment in equipment
        InvO47bea(i,j)=sum(IOmat6396(indcell{j},70,i-16)); % Column: Private fixed investment in
             intellectual property products
    end
end

indcell{24}=(28:31);
indcell{25}=(32:39);
indcell{26}=(40:43);
indcell{27}=(44:47);
indcell{28}=(48:50);
indcell{29}=(51:53);
indcell{30}=54;
indcell{31}=(55:56);
indcell{32}=57;
indcell{33}=(58:61);
indcell{34}=(62:63);
indcell{35}=64;
indcell{36}=65;
indcell{37}=66;

% BM: vLW's code has number of years hardcoded to 72
% for i=51:72
for i=51:yearnum
    for j=1:secnum
        for k=1:secnum
            %IOmat47bea(j,k,i)=sum(sum(IOmat9718(indcell{j},indcell{k},i-50)));
            IOmat47bea(j,k,i)=sum(sum(IOmat9720(indcell{j},indcell{k},i-50)));
        end
```

```
%       VA47bea(i,j)=sum(IOmat9718(82,indcell{j},i-50)); % Row: Total Value Added
%       GO47bea(i,j)=sum(IOmat9718(83,indcell{j},i-50)); % Row: Total industry Output
%       Cons47bea(i,j)=sum(sum(IOmat9718(indcell{j},[75 81],i-50))); % Column: Personal consumption expenditures,Nonresidential private fixed investment in structures, Nonresidential private fixed investment in equipment, Nonresidential private fixed investment in intellectual property products, Residential private fixed investment, Change in private inventories, Exports of goods and services
%       InvS47bea(i,j)=sum(sum(IOmat9718(indcell{j},[76 79],i-50))); % Column: Nonresidential private fixed investment in structures, Nonresidential private fixed investment in equipment, Nonresidential private fixed investment in intellectual property products, Residential private fixed investment
%       InvResbea(i,j)=sum(IOmat9718(indcell{j},79,i-50)); % Column: Residential private fixed investment
%       InvE47bea(i,j)=sum(IOmat9718(indcell{j},77,i-50)); % Column: Nonresidential private fixed investment in equipment
%       InvO47bea(i,j)=sum(IOmat9718(indcell{j},78,i-50)); % Column: Nonresidential private fixed investment in intellectual property products
        % BM: BEA has changed the data structure. Make necessary changes
        VA47bea(i,j)=sum(IOmat9720(78,indcell{j},i-50)); % Row: Total Value Added
        GO47bea(i,j)=sum(IOmat9720(79,indcell{j},i-50)); % Row: Total industry Output
        Cons47bea(i,j)=sum(sum(IOmat9720(indcell{j},[73 79],i-50))); % Column: Personal consumption expenditures,Nonresidential private fixed investment in structures, Nonresidential private fixed investment in equipment, Nonresidential private fixed investment in intellectual property products, Residential private fixed investment, Change in private inventories, Exports of goods and services
        InvS47bea(i,j)=sum(sum(IOmat9720(indcell{j},[74 77],i-50))); % Column: Nonresidential private fixed investment in structures, Nonresidential private fixed investment in equipment, Nonresidential private fixed investment in intellectual property products, Residential private fixed investment
        InvResbea(i,j)=sum(IOmat9720(indcell{j},77,i-50)); % Column: Residential private fixed investment
        InvE47bea(i,j)=sum(IOmat9720(indcell{j},75,i-50)); % Column: Nonresidential private fixed investment in equipment
        InvO47bea(i,j)=sum(IOmat9720(indcell{j},76,i-50)); % Column: Nonresidential private fixed investment in intellectual property products
    end
end

%%% Load labor share data

% BM: Change to ouput data path
cd(Dir.Output);
%labshbea = xlsread('37 Sector Data.xlsx',11,'B2:AL74'); % BM: xlsread takes 11 Sheet 10 corresponds to
```

```
    the labor share
labshbea     = table2array(readtable('37 Sector Data.xlsx','Sheet','labor_share','Range','B2:AL74'));
NominalVA47bea = table2array(readtable('37 Sector Data.xlsx','Sheet','nominal_va','Range','B2:AL74'));
RealVA47bea  = table2array(readtable('37 Sector Data.xlsx','Sheet','real_va','Range','B2:AL74'));
PriceVA47bea = table2array(readtable('37 Sector Data.xlsx','Sheet','VA_P','Range','B2:AL74'));

% BM: Change to working path
cd(Dir.Working);
labshbea=labshbea';
capshbea=1-labshbea;

%%% Load in the depreciation and nominal capital data to construct average depreciation rates
%%% by each industry (from Fixed Assets implied depreciation rate data);
%%% also used in TFP construction to construct real capital stocks

% BM: Change to input data path
cd(Dir.Input);
% 63 is number of industry for which data is available
for i=1:63
%    Inddeprecrate(i,:) = xlsread('DetailNonres_rate.xlsx',i+1,'C7:BV7');
%    Indnominv_equip(i,:) = xlsread('detailnonres_inv1.xlsx',i+1,'AW8:DP8'); % Equipment
%    Indnominv_struc(i,:) = xlsread('detailnonres_inv1.xlsx',i+1,'AW48:DP48'); % Structure
%    Indnominv_ipp(i,:) = xlsread('detailnonres_inv1.xlsx',i+1,'AW81:DP81'); % Intellectual property
    products
   % BM: Data available till 2020
   Inddeprecrate(i,:) = xlsread('DetailNonres_rate.xlsx',i+1,'C7:BX7');
   Indnominv_equip(i,:) = xlsread('detailnonres_inv1.xlsx',i+1,'AW8:DR8'); % Equipment
   Indnominv_struc(i,:) = xlsread('detailnonres_inv1.xlsx',i+1,'AW48:DR48'); % Structure
   Indnominv_ipp(i,:) = xlsread('detailnonres_inv1.xlsx',i+1,'AW81:DR81'); % Intellectual property
       products
end

% BM: Change to working path
cd(Dir.Working);
Indnominv = Indnominv_equip+Indnominv_struc+Indnominv_ipp; % Total nominal investment

% BM: Change to input data path
cd(Dir.Input);
%Indcaptot = xlsread('Nominal Year End Net Stock Capital by Industry 1947-2020.xlsx',1,'C8:BV85');
```

```
% BM: Data available till 2020
Indcaptot = xlsread('Nominal Year End Net Stock Capital by Industry 1947-2020.xlsx',1,'C8:BX85');

% BM: Change to working path
cd(Dir.Working);
Shrunkindcaptot = Indcaptot([3:4 6:10 13:23 25:34 36:43 45:48 50:54 56:57 59:62 64:66 68:71 73:74
    76:78],:);

% BM: Change to input data path
cd(Dir.Input);
% Load the non-residential capital stock for real estate
% Capequip_RE = xlsread('detailnonres_stk1.xlsx',47,'C8:BV8'); % Equipment
% Capnrstruc_RE = xlsread('detailnonres_stk1.xlsx',47,'C48:BV48'); % Structure
% Capipp_RE = xlsread('detailnonres_stk1.xlsx',47,'C81:BV81'); % Intellectual property products
%
% Capequip_RL = xlsread('detailnonres_stk1.xlsx',47,'C8:BV8'); % Same as above ?
% Capnrstruc_RL = xlsread('detailnonres_stk1.xlsx',47,'C48:BV48'); % Same as above ?
% Capipp_RL = xlsread('detailnonres_stk1.xlsx',47,'C81:BV81'); % Same as above ?

% BM: Data available till 2020
Capequip_RE = xlsread('detailnonres_stk1.xlsx',47,'C8:BX8'); % Equipment
Capnrstruc_RE = xlsread('detailnonres_stk1.xlsx',47,'C48:Bx48'); % Structure
Capipp_RE = xlsread('detailnonres_stk1.xlsx',47,'C81:BX81'); % Intellectual property products

Capequip_RL = xlsread('detailnonres_stk1.xlsx',47,'C8:BX8'); % Same as above ?
Capnrstruc_RL = xlsread('detailnonres_stk1.xlsx',47,'C48:BX48'); % Same as above ?
Capipp_RL = xlsread('detailnonres_stk1.xlsx',47,'C81:BX81'); % Same as above ?

% BM: Change to working path
cd(Dir.Working);
CapRE_nr = Capequip_RE+Capnrstruc_RE+Capipp_RE;
CapRL_nr = Capequip_RL+Capnrstruc_RL+Capipp_RL;


Depindcell{1}=(3:5);
for i=2:24
   Depindcell{i} = i+4;
end
Depindcell{25} = (29:36);
```

```
Depindcell{26} = (37:40);
Depindcell{27} = (41:45);
Depindcell{28} = (46:47);
Depindcell{29} = (48:50);
Depindcell{30} = 51;
Depindcell{31} = (52:53);
Depindcell{32} = 54;
Depindcell{33} = (55:58);
Depindcell{34} = (59:60);
Depindcell{35} = 61;
Depindcell{36} = 62;
Depindcell{37} = 63;

% BM: Change to input data path
cd(Dir.Input);
%implresiddeprec = xlsread('Constructing Residential Assets Depreciation Rate',1,'D6:BW6');
% BM: Data available till 2020
implresiddeprec = xlsread('Constructing Residential Assets Depreciation Rate',1,'D6:BY6');

% BM: Change to working path
cd(Dir.Working);
for i=1:secnum
    if i==28
        %Adjust depreciation to account for residential investment; set
        %residential depreciation based on BEA data
        % BM: Why 1000?
        depratbea(i,:)=(sum(Shrunkindcaptot(Depindcell{i},:))-CapRE_nr/1000-CapRL_nr/1000).*implresiddeprec
            ./sum(Shrunkindcaptot(Depindcell{i},:))+sum([CapRE_nr/1000;CapRL_nr/1000].*Inddeprecrate(
            Depindcell{i},:),1)./sum(Shrunkindcaptot(Depindcell{i},:),1);
    else
        depratbea(i,:)=sum(Shrunkindcaptot(Depindcell{i},:).*Inddeprecrate(Depindcell{i},:),1)./sum(
            Shrunkindcaptot(Depindcell{i},:),1);
    end
    nominvbea(i,:)=sum(Indnominv(Depindcell{i},:),1)/1000; % Nominal investment
end

% BM: Store Depreciation Rate Data
ProcessDRData = [years,depratbea'];              % Append the years as the first column
ProcessedDRData = array2table(ProcessDRData,'VariableNames',secname);
```

```
ProcessNIData = [years,nominvbea'];             % Append the years as the first column
ProcessedNIData = array2table(ProcessNIData,'VariableNames',secname);

% BM: Change to input data path
cd(Dir.Input)
writetable(ProcessedDRData,'DepreciationRate.xlsx','FileType','spreadsheet','Sheet','depreciation_rate','
    WriteMode','overwritesheet');
writetable(ProcessedNIData,'NominalInvestment.xlsx','FileType','spreadsheet','Sheet','nominal_inv','
    WriteMode','overwritesheet');

% BM: For comparison
nominvbeatanspose = nominvbea';

% BM: Change to working path
cd(Dir.Working);

% Share of Intermediates Production (for use with empirical analysis)
II47bea = reshape(sum(IOmat47bea,2),secnum,yearnum)';
IIshare = II47bea./repmat(sum(II47bea,2),1,secnum);

% BM: Alpha and Alpha_LS
GO47beamat = zeros(secnum,secnum,yearnum);
for y = 1:yearnum
   GO47beamat(:,:,y) = repmat(GO47bea(y,:),secnum,1);
end
ALPHA4720   = IOmat47bea./GO47beamat;
LS4720      = 1-sum(ALPHA4720,1);
ALPHA_LS4720 = [ALPHA4720;LS4720];
ALPHA4720   = pagetranspose(ALPHA4720);
ALPHA_LS4720 = pagetranspose(ALPHA_LS4720);
ALPHA4820   = ALPHA4720(:,:,2:end);
ALPHA_LS4820 = ALPHA_LS4720(:,:,2:end);

cd(Dir.Output);
save('ALPHA4720');
save('ALPHA_LS4720');
save('ALPHA4820');
save('ALPHA_LS4820');
```

```
cd(Dir.DataPr);
save('ALPHA4720');
save('ALPHA_LS4720');
save('ALPHA4820');
save('ALPHA_LS4820');

% BM: Change to working path
cd(Dir.Working);

% BM: Change to output data path
cd(Dir.Output);
save IOmat4720dat_37sec
cd(Dir.DataPr);
save IOmat4720dat_37sec

% BM: Change to working path
cd(Dir.Working);

% toc;
```

11.6.6 Mergedata.m

```
Input:
Input data to be merged.
Directory paths and variable types to fetch and merge corresponding data.

Output:
Merged data tables containing harmonized sector data.

Auxiliary:
Logic to handle different variable types and merge data accordingly.
Functions to fetch and process data from various sources.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bineet Mishra, September 2021
% Paper  : Endogenous Production Networks under Uncertainty
%          Kopytov, Mishra, Nimark, and Taschereau-Dumouchel
% Function: To merge the data
```

```
function MergedData = Mergedata(InputData,Dir,Var,Source)
    cd(Dir.Working);
    %  Same sector name as in 37 Sector Data.xlsx
    SecName   = {'Years','Mining','Utilities','Construction', 'WoodProducts',...
                 'NonmetallicMinerals','PrimaryMetals','FabricatedMetals',...
                 'Machinery','ComputerandElectronic', 'ElectricalEquipment',...
                 'MotorVehicles','OtherTransequip','Furnitureandrelated',...
                 'MiscMfg','Foodandbeverage','Textile','Apparel','Paper',...
                 'Printing','Petroleum','Chemical','Plastics','WT','RT','TW',...
                 'Info','FI','RE','ProfBus','Mgmt','Admin','Edu','Health',...
                 'Arts','Accomm','FoodServ','Other'};
    switch Var
       case 'RealVA'
          TableRealVA4818 = Getdata(Dir,Var,Source);
          Years4818     = TableRealVA4818.Var1;
          Year96        = find(Years4818==1996);
          RealVA4896    = TableRealVA4818{1:Year96,:};
          RealVA9720    = InputData{:,:};
          RealVA        = [RealVA4896;RealVA9720];
          MergedData    = array2table(RealVA,'VariableNames',SecName);
       case 'RealGO'
          TableRealGO4818 = Getdata(Dir,Var,Source);
          Years4818     = TableRealGO4818.Var1;
          Year96        = find(Years4818==1996);
          RealGO4896    = TableRealGO4818{1:Year96,:};
          RealGO9720    = InputData{:,:};
          RealGO        = [RealGO4896;RealGO9720];
          MergedData    = array2table(RealGO,'VariableNames',SecName);
       case 'RealII'
          TableRealII4818 = Getdata(Dir,Var,Source);
          Years4818     = TableRealII4818.Var1;
          Year96        = find(Years4818==1996);
          RealII4896    = TableRealII4818{1:Year96,:};
          RealII9720    = InputData{:,:};
          RealII        = [RealII4896;RealII9720];
          MergedData    = array2table(RealII,'VariableNames',SecName);
      case 'NominalVA'
```

```
            TableNominalVA4818 = Getdata(Dir,Var,Source);
            Years4818       = TableNominalVA4818.Var1;
            Year96          = find(Years4818==1996);
            NominalVA4896   = TableNominalVA4818{1:Year96,:};
            NominalVA9720   = InputData{:,:};
            NominalVA       = [NominalVA4896;NominalVA9720];
            MergedData      = array2table(NominalVA,'VariableNames',SecName);
        case 'NominalGO'
            TableNominalGO4818 = Getdata(Dir,Var,Source);
            Years4818       = TableNominalGO4818.Var1;
            Year96          = find(Years4818==1996);
            NominalGO4896   = TableNominalGO4818{1:Year96,:};
            NominalGO9720   = InputData{:,:};
            NominalGO       = [NominalGO4896;NominalGO9720];
            MergedData      = array2table(NominalGO,'VariableNames',SecName);
        case 'NominalII'
            TableNominalII4818 = Getdata(Dir,Var,Source);
            Years4818       = TableNominalII4818.Var1;
            Year96          = find(Years4818==1996);
            NominalII4896   = TableNominalII4818{1:Year96,:};
            NominalII9720   = InputData{:,:};
            NominalII       = [NominalII4896;NominalII9720];
            MergedData      = array2table(NominalII,'VariableNames',SecName);
        case 'Employment'
            TableEmployment4818 = Getdata(Dir,Var,Source);
            Years4818        = TableEmployment4818.Var1;
            Year97           = find(Years4818==1997);
            Employment4897   = TableEmployment4818{1:Year97,:};
            Employment9820   = InputData{:,:};
            Employment       = [Employment4897;Employment9820];
            MergedData       = array2table(Employment,'VariableNames',SecName);
        case 'LaborSharesUnScaled'
            TableLaborSharesUnScaled4818 = Getdata(Dir,Var,Source);
            Years4818                = TableLaborSharesUnScaled4818.Var1;
            Year96                   = find(Years4818==1996);
            LaborSharesUnScaled4896  = TableLaborSharesUnScaled4818{1:Year96,:};
            LaborSharesUnScaled9720  = InputData{:,:};
            LaborSharesUnScaled      = [LaborSharesUnScaled4896;LaborSharesUnScaled9720];
            MergedData               = array2table(LaborSharesUnScaled,'VariableNames',SecName);
```

```
        case 'NominalInvestment'
            TableNominalInvestment4720 = Getdata(Dir,Var,Source);
            IndexRE                    = find(contains(TableNominalInvestment4720.Var2,'Real estate and
                rental and leasing'));
            NIRE                       = TableNominalInvestment4720{IndexRE,4:end}';
            InputData.RE               = NIRE;
            NominalInvestment          = InputData;
            MergedData                 = NominalInvestment;
        otherwise
            fprintf('Incorrect variable. Set Var from any one of the following: \n RealVA \n RealGO \n
                RealII \n NominalVA \n NominalGO \n NominalII \n IIShare \n Employment \n
                LaborSharesUnScaled \n SelfEmployed \n ScalingFactor \n LaborShares \n NominalCapital \n
                DepreciationRate \n NominalInvestment \n RealInvestment \n RealInvestmentDollars \n');
    end
end
```

11.6.7 Processdata.m

```
Input:
Input data to be processed.
Directory paths and variable types to process corresponding data.

Output:
Processed data tables containing various economic indicators.

Auxiliary:
Logic to handle different variable types and process data accordingly.
Functions to fetch and process data from various sources.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bineet Mishra, September 2021
% Paper  : Endogenous Production Networks under Uncertainty
%          Kopytov, Mishra, Nimark, and Taschereau-Dumouchel
% Function: To process the data


function ProcessedData = Processdata(InputData,Dir,Var)
    cd(Dir.Working);
```

```matlab
% BEA has data for different time frames: Change StartYear and LastYear
% according to the data downloaded
StartYear = 1997;
LastYear  = 2020;
Secnum    = 37;                       % Number of sectors
Yearnum   = LastYear-StartYear+1;     % Number of years
Years     = (StartYear:1:LastYear)';  % Years
%  Same sector name as in 37 Sector Data.xlsx
SecName   = {'Years','Mining','Utilities','Construction', 'WoodProducts',...
             'NonmetallicMinerals','PrimaryMetals','FabricatedMetals',...
             'Machinery','ComputerandElectronic', 'ElectricalEquipment',...
             'MotorVehicles','OtherTransequip','Furnitureandrelated',...
             'MiscMfg','Foodandbeverage','Textile','Apparel','Paper',...
             'Printing','Petroleum','Chemical','Plastics','WT','RT','TW',...
             'Info','FI','RE','ProfBus','Mgmt','Admin','Edu','Health',...
             'Arts','Accomm','FoodServ','Other'};
switch Var
   case 'RealVA'
      RealVAData = InputData{1:end,3:end}'; % Real gross output data
      RealVAData(isnan(RealVAData)) = 0;  % Convert NaN observations (....) in the Excel files to zero
      % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
      % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
      Seccell   = cell(Secnum,1);
      Seccell{1} = 6; % Mining
      % Utilities,Construction
      for s = 2:3
         Seccell{s} = s+8;
      end
      % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
      % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
      % Furnitureandrelated,MiscMfg
      for s = 4:14
         Seccell{s} = s+10;
      end
      % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT,RT
      for s = 15:24
         Seccell{s} = s+11;
      end
```

```
        Seccell{25} = 40; % TW
        Seccell{26} = 49; % Info
        Seccell{27} = 55; % FI
        Seccell{28} = 60; % RE
        Seccell{29} = 66; % ProfBus
        Seccell{30} = 70; % Mgmt
        Seccell{31} = 71; % Admin
        Seccell{32} = 75; % Edu
        Seccell{33} = 76; % Health
        Seccell{34} = 82; % Arts
        Seccell{35} = 86; % Accomm
        Seccell{36} = 87; % FoodServ
        Seccell{37} = 88; % Other
        ProcessData = zeros(Yearnum,Secnum);       % Preallocate the matrix
        for s = 1:Secnum
            ProcessData(:,s) = RealVAData(:,Seccell{s}); % Collect relevant 37 sector data
        end
        ProcessData = [Years,ProcessData];         % Append the years as the first column
        ProcessedData = array2table(ProcessData,'VariableNames',SecName);
    case 'RealGO'
        RealGOData = InputData{1:end,3:end}'; % Real gross output data
        RealGOData(isnan(RealGOData)) = 0;  % Convert NaN observations (....) in the Excel files to zero
        % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
        % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
        Seccell   = cell(Secnum,1);
        Seccell{1} = 6; % Mining
        % Utilities,Construction
        for s = 2:3
            Seccell{s} = s+8;
        end
        % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
        % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
        % Furnitureandrelated,MiscMfg
        for s = 4:14
            Seccell{s} = s+10;
        end
        % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT,RT
        for s = 15:24
```

```
                Seccell{s} = s+11;
            end
            Seccell{25} = 40; % TW
            Seccell{26} = 49; % Info
            Seccell{27} = 55; % FI
            Seccell{28} = 60; % RE
            Seccell{29} = 66; % ProfBus
            Seccell{30} = 70; % Mgmt
            Seccell{31} = 71; % Admin
            Seccell{32} = 75; % Edu
            Seccell{33} = 76; % Health
            Seccell{34} = 82; % Arts
            Seccell{35} = 86; % Accomm
            Seccell{36} = 87; % FoodServ
            Seccell{37} = 88; % Other
            ProcessData = zeros(Yearnum,Secnum);       % Preallocate the matrix
            for s = 1:Secnum
                ProcessData(:,s) = RealGOData(:,Seccell{s}); % Collect relevant 37 sector data
            end
            ProcessData = [Years,ProcessData];         % Append the years as the first column
            ProcessedData = array2table(ProcessData,'VariableNames',SecName);
        case 'RealII'
            RealIIData = InputData{1:end,3:end}'; % Real intermediate inputs data
            RealIIData(isnan(RealIIData)) = 0;  % Convert NaN observations (....) in the Excel files to zero
            % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
            % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
            Seccell   = cell(Secnum,1);
            Seccell{1} = 6; % Mining
            % Utilities,Construction
            for s = 2:3
                Seccell{s} = s+8;
            end
            % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
            % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
            % Furnitureandrelated,MiscMfg
            for s = 4:14
                Seccell{s} = s+10;
            end
```

```
        % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT,RT
        for s = 15:24
           Seccell{s} = s+11;
        end
        Seccell{25} = 40; % TW
        Seccell{26} = 49; % Info
        Seccell{27} = 55; % FI
        Seccell{28} = 60; % RE
        Seccell{29} = 66; % ProfBus
        Seccell{30} = 70; % Mgmt
        Seccell{31} = 71; % Admin
        Seccell{32} = 75; % Edu
        Seccell{33} = 76; % Health
        Seccell{34} = 82; % Arts
        Seccell{35} = 86; % Accomm
        Seccell{36} = 87; % FoodServ
        Seccell{37} = 88; % Other
        ProcessData = zeros(Yearnum,Secnum);       % Preallocate the matrix
        for s = 1:Secnum
           ProcessData(:,s) = RealIIData(:,Seccell{s}); % Collect relevant 37 sector data
        end
        ProcessData = [Years,ProcessData];         % Append the years as the first column
        ProcessedData = array2table(ProcessData,'VariableNames',SecName);
     case 'NominalVA'
        NominalVAData = InputData{1:end,3:end}'; % Nominal gross output data
        NominalVAData(isnan(NominalVAData)) = 0; % Convert NaN observations (....) in the Excel files to zero
        % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
        % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
        Seccell   = cell(Secnum,1);
        Seccell{1} = 6; % Mining
        % Utilities,Construction
        for s = 2:3
           Seccell{s} = s+8;
        end
        % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
        % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
        % Furnitureandrelated,MiscMfg
        for s = 4:14
```

```
            Seccell{s} = s+10;
        end
        % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT,RT
        for s = 15:24
            Seccell{s} = s+11;
        end
        Seccell{25} = 40; % TW
        Seccell{26} = 49; % Info
        Seccell{27} = 55; % FI
        Seccell{28} = 60; % RE
        Seccell{29} = 66; % ProfBus
        Seccell{30} = 70; % Mgmt
        Seccell{31} = 71; % Admin
        Seccell{32} = 75; % Edu
        Seccell{33} = 76; % Health
        Seccell{34} = 82; % Arts
        Seccell{35} = 86; % Accomm
        Seccell{36} = 87; % FoodServ
        Seccell{37} = 88; % Other
        ProcessData = zeros(Yearnum,Secnum);         % Preallocate the matrix
        for s = 1:Secnum
            ProcessData(:,s) = NominalVAData(:,Seccell{s}); % Collect relevant 37 sector data
        end
        ProcessData = [Years,ProcessData];           % Append the years as the first column
        ProcessedData = array2table(ProcessData,'VariableNames',SecName);
    case 'NominalGO'
        NominalGOData = InputData{1:end,3:end}'; % Nominal gross output data
        NominalGOData(isnan(NominalGOData)) = 0; % Convert NaN observations (....) in the Excel files to zero
        % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
        % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
        Seccell   = cell(Secnum,1);
        Seccell{1} = 6; % Mining
        % Utilities,Construction
        for s = 2:3
            Seccell{s} = s+8;
        end
        % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
        % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
```

```
        % Furnitureandrelated,MiscMfg
        for s = 4:14
            Seccell{s} = s+10;
        end
        % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT,RT
        for s = 15:24
            Seccell{s} = s+11;
        end
        Seccell{25} = 40; % TW
        Seccell{26} = 49; % Info
        Seccell{27} = 55; % FI
        Seccell{28} = 60; % RE
        Seccell{29} = 66; % ProfBus
        Seccell{30} = 70; % Mgmt
        Seccell{31} = 71; % Admin
        Seccell{32} = 75; % Edu
        Seccell{33} = 76; % Health
        Seccell{34} = 82; % Arts
        Seccell{35} = 86; % Accomm
        Seccell{36} = 87; % FoodServ
        Seccell{37} = 88; % Other
        ProcessData = zeros(Yearnum,Secnum);          % Preallocate the matrix
        for s = 1:Secnum
            ProcessData(:,s) = NominalGOData(:,Seccell{s}); % Collect relevant 37 sector data
        end
        ProcessData = [Years,ProcessData];            % Append the years as the first column
        ProcessedData = array2table(ProcessData,'VariableNames',SecName);
    case 'NominalII'
        NominalIIData = InputData{1:end,3:end}'; % Nominal intermediate inputs data
        NominalIIData(isnan(NominalIIData)) = 0; % Convert NaN observations (....) in the Excel files to zero
        % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
        % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
        Seccell   = cell(Secnum,1);
        Seccell{1} = 6; % Mining
        % Utilities,Construction
        for s = 2:3
            Seccell{s} = s+8;
        end
```

```
      % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
      % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
      % Furnitureandrelated,MiscMfg
      for s = 4:14
         Seccell{s} = s+10;
      end
      % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT,RT
      for s = 15:24
         Seccell{s} = s+11;
      end
      Seccell{25} = 40; % TW
      Seccell{26} = 49; % Info
      Seccell{27} = 55; % FI
      Seccell{28} = 60; % RE
      Seccell{29} = 66; % ProfBus
      Seccell{30} = 70; % Mgmt
      Seccell{31} = 71; % Admin
      Seccell{32} = 75; % Edu
      Seccell{33} = 76; % Health
      Seccell{34} = 82; % Arts
      Seccell{35} = 86; % Accomm
      Seccell{36} = 87; % FoodServ
      Seccell{37} = 88; % Other
      ProcessData = zeros(Yearnum,Secnum);          % Preallocate the matrix
      for s = 1:Secnum
         ProcessData(:,s) = NominalIIData(:,Seccell{s}); % Collect relevant 37 sector data
      end
      ProcessData = [Years,ProcessData];            % Append the years as the first column
      ProcessedData = array2table(ProcessData,'VariableNames',SecName);
   case 'IIShare'
      NominalII   = InputData.NominalII;
      NominalGO   = InputData.NominalGO;
      ProcessData = NominalII{:,2:end}./NominalGO{:,2:end}; % Shares = NominalII/NominalGO
      Years       = NominalII.Years;
      ProcessData = [Years,ProcessData];               % Append the years as the first column
      ProcessedData = array2table(ProcessData,'VariableNames',SecName);
   case 'Employment'
      % BEA has data for different time frames: Change StartYear and LastYear
      % according to the data downloaded
```

```
        StartYear = 1998;
        LastYear  = 2020;
        Yearnum   = LastYear-StartYear+1;  % Number of years
        Years     = (StartYear:1:LastYear)'; % Years
        EmploymentData = InputData{1:end,3:end}';   % Employment data
        EmploymentData(isnan(EmploymentData)) = 0;  % Convert NaN observations (....) in the Excel
            files to zero
        % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
        % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
        Seccell   = cell(Secnum,1);
        Seccell{1} = 7; % Mining
        % Utilities,Construction
        for s = 2:3
           Seccell{s} = s+9;
        end
        % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
        % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
        % Furnitureandrelated,MiscMfg
        for s = 4:14
           Seccell{s} = s+11;
        end
        % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT
        for s = 15:23
           Seccell{s} = s+12;
        end
        Seccell{24} = 38; % RT
        Seccell{25} = 43; % TW
        Seccell{26} = 52; % Info
        Seccell{27} = 57; % FI
        Seccell{28} = 62; % RE
        Seccell{29} = 65; % ProfBus
        Seccell{30} = 69; % Mgmt
        Seccell{31} = 70; % Admin
        Seccell{32} = 73; % Edu
        Seccell{33} = 74; % Health
        Seccell{34} = 79; % Arts
        Seccell{35} = 83; % Accomm
        Seccell{36} = 84; % FoodServ
        Seccell{37} = 85; % Other
```

```
            ProcessData = zeros(Yearnum,Secnum);            % Preallocate the matrix
            for s = 1:Secnum
                ProcessData(:,s) = EmploymentData(:,Seccell{s}); % Collect relevant 37 sector data
            end
            ProcessData = [Years,ProcessData];              % Append the years as the first column
            ProcessedData = array2table(ProcessData,'VariableNames',SecName);
        case 'LaborSharesUnScaled'
            % BEA has data for different time frames: Change StartYear and LastYear
            % according to the data downloaded
            VACompData = InputData{1:end,3:end}; % Value added components data
            VACompData(isnan(VACompData)) = 0;  % Convert NaN observations (....) in the Excel files to zero
            Secline   = 388;
            s = 2:4:Secline;
            SecComp   = VACompData(s,:);
            SecVA     = VACompData(s-1,:);
            SecTax    = VACompData(s+1,:);
            SecLS     = SecComp./(SecVA-SecTax);
            SecLSData = SecLS';
            % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
            % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
            Seccell   = cell(Secnum,1);
            Seccell{1} = 6; % Mining
            % Utilities,Construction
            for s = 2:3
                Seccell{s} = s+8;
            end
            % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
            % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
            % Furnitureandrelated,MiscMfg
            for s = 4:14
                Seccell{s} = s+10;
            end
            % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT,RT
            for s = 15:24
                Seccell{s} = s+11;
            end
            Seccell{25} = 40; % TW
            Seccell{26} = 49; % Info
```

```
        Seccell{27} = 55; % FI
        Seccell{28} = 60; % RE
        Seccell{29} = 66; % ProfBus
        Seccell{30} = 70; % Mgmt
        Seccell{31} = 71; % Admin
        Seccell{32} = 75; % Edu
        Seccell{33} = 76; % Health
        Seccell{34} = 82; % Arts
        Seccell{35} = 86; % Accomm
        Seccell{36} = 87; % FoodServ
        Seccell{37} = 88; % Other
        ProcessData = zeros(Yearnum,Secnum);      % Preallocate the matrix
        for s = 1:Secnum
           ProcessData(:,s) = SecLSData(:,Seccell{s}); % Collect relevant 37 sector data
        end
        ProcessData = [Years,ProcessData];        % Append the years as the first column
        ProcessedData = array2table(ProcessData,'VariableNames',SecName);
     case 'SelfEmployed'
        % 1998-2020
        % BEA has data for different time frames: Change StartYear and LastYear
        % according to the data downloaded
        StartYear = 1998;
        LastYear  = 2020;
        Years     = (StartYear:1:LastYear)';      % Years
        SelfEmployedData9820 = InputData.SE9820{1:end,3:end}'; % Self employment data
        SelfEmployedData9820(isnan(SelfEmployedData9820)) = 0; % Convert NaN observations (....) in the
             Excel files to zero
        Secnum    = 14;                        % Number of sectors
        Yearnum   = LastYear-StartYear+1;     % Number of years
        % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
        % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
        Seccell   = cell(Secnum,1);
        % Mining, Utilities,Construction
        for s = 1:3
           Seccell{s} = s+4;
        end
        % Dur Mfg, Nondur Mfg, WT, RT,Transp, Info, FIRE, Prof/Bus, Ed/Health,
        % Arts/Ent/Accomm/Food, Other Serv
        for s = 4:14
```

```
            Seccell{s} = s+5;
        end
        ProcessData9820 = zeros(Yearnum,Secnum);           % Preallocate the matrix
        for s = 1:Secnum
            ProcessData9820(:,s) = SelfEmployedData9820(:,Seccell{s}); % Collect relevant 37 sector data
        end
        ProcessData9820 = [Years,ProcessData9820];            % Append the years as the first column
        % 1987-1997
        % BEA has data for different time frames: Change StartYear and LastYear
        % according to the data downloaded
        StartYear = 1987;
        LastYear  = 1997;
        Years     = (StartYear:1:LastYear)';  % Years
        SelfEmployedData8797 = InputData.SE8797{1:end,3:end}'; % Self employment data
        SelfEmployedData8797(isnan(SelfEmployedData8797)) = 0; % Convert NaN observations (....) in the
             Excel files to zero
        Secnum    = 14;                        % Number of sectors
        Yearnum   = LastYear-StartYear+1;      % Number of years
        % % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
        % % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
        Seccell   = cell(Secnum,1);
        Seccell{1} = 5; % Mining,
        Seccell{3} = 6; % Construction
        % Dur Mfg, Nondur Mfg
        for s = 4:5
            Seccell{s} = s+4;
        end
        % WT, RT
        for s = 6:7
            Seccell{s} = s+5;
        end
        % Transp
        Seccell{8} = 10;
        % FIRE
        Seccell{10} = 13;
        ProcessData8797 = zeros(Yearnum,Secnum);      % Preallocate the matrix
        for s = 1:Secnum
            if ~isempty(Seccell{s})
                ProcessData8797(:,s) = SelfEmployedData8797(:,Seccell{s}); % Collect relevant 37 sector
```

```
                  data
              end
          end
          AllServices8797 = SelfEmployedData8797(:,14);
          % Info, Prof/Bus, Ed/Health, Arts/Ent/Accomm/Food, Other Serv
          Services9820 = [ProcessData9820(:,10),ProcessData9820(:,12:15)];
          ServicesShare9820 = Services9820(:,:)./sum(Services9820(:,:),2);
          ServicesShareAvg9820 = mean(ServicesShare9820);
          Services8797 = ServicesShareAvg9820.*AllServices8797;
          ProcessData8797(:,9) = Services8797(:,1);
          ProcessData8797(:,11:14) = Services8797(:,2:end);
          ProcessData8797 = [Years,ProcessData8797];             % Append the years as the first column
          ProcessData = [ProcessData8797;ProcessData9820];
          %  Same sector name as in 37 Sector Data.xlsx
          SecName = {'Years','Mining','Utilities','Construction','Dur Mfg',...
                   'Nondur Mfg','WT','RT','Transp','Info','FIRE','Prof/Bus',...
                   'Ed/Health','Arts/Ent/Accomm/Food','Other Serv'};
          ProcessedData = array2table(ProcessData,'VariableNames',SecName);
      case 'ScalingFactor'
          Employment = InputData.Employment{:,2:end};
          SelfEmployed = InputData.SelfEmployed{:,2:end};
          Yearnum    = size(SelfEmployed,1);
          Secnumse   = size(SelfEmployed,2);
          Employment14 = zeros(Yearnum,Secnumse);
          Seccell    = cell(Secnumse,1);
          Seccell{1} = 1;
          Employment14(:,1) = Employment(:,Seccell{1});
          Seccell{2} = 2;
          Employment14(:,2) = Employment(:,Seccell{2});
          Seccell{3} = 3;
          Employment14(:,3) = Employment(:,Seccell{3});
          Seccell{4} = (4:14);
          Employment14(:,4) = sum(Employment(:,Seccell{4}),2);
          Seccell{5} = (15:22);
          Employment14(:,5) = sum(Employment(:,Seccell{5}),2);
          Seccell{6} = 23;
          Employment14(:,6) = Employment(:,Seccell{6});
          Seccell{7} = 24;
          Employment14(:,7) = Employment(:,Seccell{7});
```

```
        Seccell{8} = 25;
        Employment14(:,8) = Employment(:,Seccell{8});
        Seccell{9} = 26;
        Employment14(:,9) = Employment(:,Seccell{9});
        Seccell{10} = (27:28);
        Employment14(:,10) = sum(Employment(:,Seccell{10}),2);
        Seccell{11} = [29,31];
        Employment14(:,11) = sum(Employment(:,Seccell{11}),2);
        Seccell{12} = (32:33);
        Employment14(:,12) = sum(Employment(:,Seccell{12}),2);
        Seccell{13} = (34:36);
        Employment14(:,13) = sum(Employment(:,Seccell{13}),2);
        Seccell{14} = 37;
        Employment14(:,14) = Employment(:,Seccell{14});
        ScalingFactor14Actual = SelfEmployed./Employment14;
        Secnum     = size(Employment,2);
        ScalingFactor37Actual = zeros(Yearnum,Secnum);
        for s = 1:14
           if s == 4
              ScalingFactor37Actual(:,Seccell{s}) = repmat(ScalingFactor14Actual(:,s),1,11);
           elseif s == 5
              ScalingFactor37Actual(:,Seccell{s}) = repmat(ScalingFactor14Actual(:,s),1,8);
           elseif s == 10
              ScalingFactor37Actual(:,Seccell{s}) = repmat(ScalingFactor14Actual(:,s),1,2);
           elseif s == 11
              ScalingFactor37Actual(:,Seccell{s}) = repmat(ScalingFactor14Actual(:,s),1,2);
           elseif s == 12
              ScalingFactor37Actual(:,Seccell{s}) = repmat(ScalingFactor14Actual(:,s),1,2);
           elseif s == 13
              ScalingFactor37Actual(:,Seccell{s}) = repmat(ScalingFactor14Actual(:,s),1,3);
           else
              ScalingFactor37Actual(:,Seccell{s}) = ScalingFactor14Actual(:,s);
           end
        end
        ProcessData = mean(ScalingFactor37Actual);
        SecName     = SecName(2:end);                              % No need of years as scaling fator is average over the time dimension
        ProcessedData = array2table(ProcessData,'VariableNames',SecName);
     case 'LaborShares'
```

```
        LaborShareUnScaled = InputData.LaborShareUS{:,2:end};
        ScalingFactor    = InputData.ScalingFactor{1,:};
        ProcessData      = LaborShareUnScaled.*(1+ScalingFactor);
        StartYear        = 1948;
        LastYear         = 2020;
        Years            = (StartYear:1:LastYear)';   % Years
        ProcessData      = [Years,ProcessData];       % Append the years as the first column
        ProcessedData    = array2table(ProcessData,'VariableNames',SecName);
    case 'NominalCapital'
        % BEA has data for different time frames: Change StartYear and LastYear
        % according to the data downloaded
        StartYear = 1948;
        LastYear  = 2020;
        Years     = (StartYear:1:LastYear)';          % Years
        NominalCapitalData = InputData{1:end,4:end}'; % Nominal capital data
        NominalCapitalData(isnan(NominalCapitalData)) = 0; % Convert NaN observations (....) in the Excel files to zero
        Secnum    = 37;
        Yearnum   = (LastYear-StartYear+1);
        % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
        % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
        Seccell   = cell(Secnum,1);
        Seccell{1} = 5; % Mining
        % Utilities,Construction
        for s = 2:3
           Seccell{s} = s+7;
        end
        % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
        % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
        % Furnitureandrelated,MiscMfg
        for s = 4:14
           Seccell{s} = s+9;
        end
        % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT,RT,TW
        for s = 15:25
           Seccell{s} = s+10;
        end
        Seccell{26} = 44; % Info
        Seccell{27} = 49; % FI
```

```
        Seccell{28} = 55; % RE
        Seccell{29} = 58; % ProfBus
        Seccell{30} = 62; % Mgmt
        Seccell{31} = 63; % Admin
        Seccell{32} = 66; % Edu
        Seccell{33} = 67; % Health
        Seccell{34} = 72; % Arts
        Seccell{35} = 76; % Accomm
        Seccell{36} = 77; % FoodServ
        Seccell{37} = 78; % Other
        ProcessData = zeros(Yearnum,Secnum);              % Preallocate the matrix
        for s = 1:37
           ProcessData(:,s) = NominalCapitalData(:,Seccell{s}); % Collect relevant 37 sector data
        end
        ProcessData = [Years,ProcessData];                % Append the years as the first column
        ProcessedData = array2table(ProcessData,'VariableNames',SecName);
     case 'DepreciationRate'
        ProcessedData = InputData(2:end,:);
     case 'NominalInvestment'
        ProcessedData = InputData(2:end,:);
     case 'RealInvestment'
        % BEA has data for different time frames: Change StartYear and LastYear
        % according to the data downloaded
        StartYear = 1948;
        LastYear  = 2020;
        Years     = (StartYear:1:LastYear)';         % Years
        RealInvestmentData = InputData{1:end,4:end}'; % Real investment data
        RealInvestmentData(isnan(RealInvestmentData)) = 0; % Convert NaN observations (....) in the
            Excel files to zero
        Secnum    = 37;
        Yearnum   = (LastYear-StartYear+1);
        % Same logic as in 'LoadIO_37sec.m' code of Lehn and Winberry (2020)
        % Mapping the BEA sectors to be consistent in 37 Sector Data.xlsx
        Seccell   = cell(Secnum,1);
        Seccell{1} = 5; % Mining
        % Utilities,Construction
        for s = 2:3
           Seccell{s} = s+7;
        end
```

```
            % WoodProducts,NonmetallicMinerals,PrimaryMetals,FabricatedMetals,Machinery,
            % ComputerandElectronic,ElectricalEquipment,MotorVehicles,OtherTransequip,
            % Furnitureandrelated,MiscMfg
            for s = 4:14
                Seccell{s} = s+9;
            end
            % Foodandbeverage,Textile,Apparel,Paper,Printing,Petroleum,Chemical,Plastics,WT,RT,TW
            for s = 15:25
                Seccell{s} = s+10;
            end
            Seccell{26} = 44; % Info
            Seccell{27} = 49; % FI
            Seccell{28} = 55; % RE
            Seccell{29} = 58; % ProfBus
            Seccell{30} = 62; % Mgmt
            Seccell{31} = 63; % Admin
            Seccell{32} = 66; % Edu
            Seccell{33} = 67; % Health
            Seccell{34} = 72; % Arts
            Seccell{35} = 76; % Accomm
            Seccell{36} = 77; % FoodServ
            Seccell{37} = 78; % Other
            ProcessData = zeros(Yearnum,Secnum);              % Preallocate the matrix
            for s = 1:37
                ProcessData(:,s) = RealInvestmentData(:,Seccell{s}); % Collect relevant 37 sector data
            end
            ProcessData = [Years,ProcessData];                % Append the years as the first column
            ProcessedData = array2table(ProcessData,'VariableNames',SecName);
        case 'RealInvestmentDollars'
            NominalInvestment = InputData.NominalInvestment{:,2:end};
            RealInvestment  = InputData.RealInvestment{:,2:end};
            RealInvDollar(1,:) = NominalInvestment(1,:);
            Yearnum = size(NominalInvestment,1);
            for s = 2:Yearnum
                RealInvDollar(s,:) = RealInvDollar(s-1,:).*exp(log(RealInvestment(s,:)./RealInvestment(s
                    -1,:)));
            end
            Years           = InputData.NominalInvestment{:,1};
            ProcessData     = [Years,RealInvDollar];      % Append the years as the first column
```

```
        ProcessedData   = array2table(ProcessData,'VariableNames',SecName);
    case 'VAPrice'
        BaseYear       = 2009;
        NominalVA      = InputData.NominalVA{:,2:end};
        RealVA         = InputData.RealVA{:,2:end};
        Years          = InputData.NominalVA{:,1};
        Yearindex      = find(Years == BaseYear);
        PriceVABaseYear = NominalVA(Yearindex,:)./RealVA(Yearindex,:);
        PriceVA        = ((NominalVA./RealVA)./(PriceVABaseYear)).*100;
        ProcessData    = [Years,PriceVA];
        ProcessedData  = array2table(ProcessData,'VariableNames',SecName);
    case 'TFP'
        real_GO          = InputData.real_GO{:,2:end};
        real_II          = InputData.real_II{:,2:end};
        II_shares        = InputData.II_shares{:,2:end};
        employment       = InputData.employment{:,2:end};
        labor_share      = InputData.labor_share{:,2:end};
        labor_share(find(labor_share>0.95)) = 0.95;
        nominal_capital  = InputData.nominal_capital{:,2:end};
        depreciation_rates = InputData.depreciation_rates{:,2:end};
        real_inv_dollars = InputData.real_inv_dollars{:,2:end};

        StartYear = 1948;
        LastYear  = 2020;
        Secnum    = 37;                      % Number of sectors
        Yearnum   = LastYear-StartYear+1;    % Number of years
        Years     = (StartYear:1:LastYear)'; % Years

        % Capital
        capital(1,:)      = nominal_capital(1,:);
        for t = 2:Yearnum
            capital(t,:)   = (1-depreciation_rates(t,:)).*capital(t-1,:)+ real_inv_dollars(t,:);
        end

        % Average labor share
        avg_labor_share    = movmean(labor_share,2,1);
        avg_labor_share_sm = mean(labor_share,1);

        % Average II share
```

```
        avg_II_share      = movmean(II_shares,2,1);
        avg_II_shares_sm  = mean(II_shares,1);

        % Solow residuals from gross output identity
        dtfp_go           = log(real_GO(2:end,:)./real_GO(1:end-1,:)) ...
                          - ((1-avg_II_share(2:end,:)).*avg_labor_share(2:end,:).*log(employment(2:end
                             ,:)./(employment(1:end-1,:))))...
                          - ((1-avg_II_share(2:end,:)).*(1-avg_labor_share(2:end,:)).*log(capital(2:end
                             ,:)./(capital(1:end-1,:))))...
                          - ((avg_II_share(2:end,:)).*log(real_II(2:end,:)./(real_II(1:end-1,:))));
        dtfp_go           = [ones(1,Secnum);dtfp_go];

        TFP_GO            = ones(Yearnum,Secnum);

        for t = 2:Yearnum
            TFP_GO(t,:)    = TFP_GO(t-1,:).*exp(dtfp_go(t,:));
        end

        % Solow residuals from gross output identity (smooth)
        dtfp_go_sm        = log(real_GO(2:end,:)./real_GO(1:end-1,:)) ...
                          - ((1-avg_II_shares_sm).*avg_labor_share_sm.*log(employment(2:end,:)./(
                             employment(1:end-1,:))))...
                          - ((1-avg_II_shares_sm).*(1-avg_labor_share_sm).*log(capital(2:end,:)./(
                             capital(1:end-1,:))))...
                          - ((avg_II_shares_sm).*log(real_II(2:end,:)./(real_II(1:end-1,:))));
        dtfp_go_sm        = [ones(1,Secnum);dtfp_go_sm];

        TFP_GO_sm         = ones(Yearnum,Secnum);

        for t = 2:Yearnum
            TFP_GO_sm(t,:) = TFP_GO_sm(t-1,:).*exp(dtfp_go_sm(t,:));
        end

        % Solow residuals from gross output identity (non smooth)
        dtfp_go_non_smooth = log(real_GO(2:end,:)./real_GO(1:end-1,:)) ...
                          - ((1-II_shares(2:end,:)).*labor_share(2:end,:).*log(employment(2:end,:)) -
                             ...
                           (1-II_shares(1:end-1,:)).*labor_share(1:end-1,:).*log(employment(1:end-1,:))
                              )...
```

```
                            - ((1-II_shares(2:end,:)).*(1-labor_share(2:end,:)).*log(capital(2:end,:)) - ...
                             (1-II_shares(1:end-1,:)).*(1-labor_share(1:end-1,:)).*log(capital(1:end-1,:)))...
                            - ((II_shares(2:end,:)).*log(real_II(2:end,:))-...
                             (II_shares(1:end-1,:)).*log(real_II(1:end-1,:)));
          dtfp_go_non_smooth = [ones(1,Secnum);dtfp_go_non_smooth];
          TFP_GO_nsm         = ones(Yearnum,Secnum);
          for t = 2:Yearnum
              TFP_GO_nsm(t,:) = TFP_GO_nsm(t-1,:).*exp(dtfp_go_non_smooth(t,:));
          end

          % Solow residuals from gross output identity (non smooth not normalized)
          TFP_GO_nsm_nn      = real_GO./...
                             ((employment.^(labor_share).* ...
                             capital.^(1-labor_share)).^(1-II_shares).* ...
                             (real_II.^(II_shares)));

          ProcessDataTG      = [Years,TFP_GO];                    % Append the years as the first column
          ProcessDataTGS     = [Years,TFP_GO_sm];                 % Append the years as the first column
          ProcessDataTGNS    = [Years,TFP_GO_nsm];                % Append the years as the first column
          ProcessDataTGNSNN = [Years,TFP_GO_nsm_nn];              % Append the years as the first column
          TGTable            = array2table(ProcessDataTG,'VariableNames',SecName);
          TGSTable           = array2table(ProcessDataTGS,'VariableNames',SecName);
          TGNSTable          = array2table(ProcessDataTGNS,'VariableNames',SecName);
          TGNSNNTable        = array2table(ProcessDataTGNSNN,'VariableNames',SecName);
          ProcessedData.TG   = TGTable;
          ProcessedData.TGS  = TGSTable;
          ProcessedData.TGNS = TGNSTable;
          ProcessedData.TGNSNN = TGNSNNTable;
      otherwise
          fprintf('Incorrect variable. Set Var from any one of the following: \n RealVA \n RealGO \n RealII \n NominalVA \n NominalGO \n NominalII \n IIShare \n Employment \n LaborSharesUnScaled \n SelfEmployed \n ScalingFactor \n LaborShares \n NominalCapital \n DepreciationRate \n NominalInvestment \n RealInvestment \n RealInvestmentDollars \n TFP \n');
  end
end
```

11.6.8 Storedata.m

```
Input:
Processed data to be stored.
Directory paths and variable types to store corresponding data.

Output:
Status indicating whether the data storage was successful or not.

Auxiliary:
Functions to store data in specified Excel files and sheets.
Logic to handle different variable types and store data accordingly.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bineet Mishra, September 2021
% Paper  : Endogenous Production Networks under Uncertainty
%          Kopytov, Mishra, Nimark, and Taschereau-Dumouchel
% Function: To store the file in relevant location

function StatusPr = Storedata(ProcessedData,Dir,Var)
    cd(Dir.Output);
    switch Var
        case 'RealVA'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','real_va','
                WriteMode','overwritesheet');
        case 'RealGO'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','real_GO','
                WriteMode','overwritesheet');
        case 'RealII'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','real_II','
                WriteMode','overwritesheet');
        case 'NominalVA'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','nominal_va','
                WriteMode','overwritesheet');
        case 'NominalGO'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','nominal_GO','
                WriteMode','overwritesheet');
        case 'NominalII'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','nominal_II','
```

```
            WriteMode','overwritesheet');
        case 'IIShare'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','II_shares','WriteMode','overwritesheet');
        case 'Employment'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','employment','WriteMode','overwritesheet');
        case 'LaborSharesUnScaled'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','labor_share_unscaled','WriteMode','overwritesheet');
        case 'SelfEmployed'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','selfemployed','WriteMode','overwritesheet');
        case 'ScalingFactor'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','scalingfactor','WriteMode','overwritesheet');
        case 'LaborShares'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','labor_share','WriteMode','overwritesheet');
        case 'NominalCapital'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','nominal_capital','WriteMode','overwritesheet');
        case 'DepreciationRate'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','depreciation_rates','WriteMode','overwritesheet');
        case 'NominalInvestment'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','nominal_inv','WriteMode','overwritesheet');
        case 'RealInvestment'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','real_inv','WriteMode','overwritesheet');
        case 'RealInvestmentDollars'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','real_inv_dollars','WriteMode','overwritesheet');
        case 'VAPrice'
            writetable(ProcessedData,'37 Sector Data.xlsx','FileType','spreadsheet','Sheet','VA_P','WriteMode','overwritesheet');
        case 'TFP'
            TG    = ProcessedData.TG;
```

```
        TGS    = ProcessedData.TGS;
        TGNS   = ProcessedData.TGNS;
        TGNSNN = ProcessedData.TGNSNN;
        writetable(TG,'TFP_GO.xlsx','FileType','spreadsheet','Sheet','TFP_GO','WriteMode','
            overwritesheet');
        writetable(TGS,'TFP_GO_sm.xlsx','FileType','spreadsheet','Sheet','TFP_GO_sm','WriteMode','
            overwritesheet');
        writetable(TGNS,'TFP_GO_nsm.xlsx','FileType','spreadsheet','Sheet','TFP_GO_nsm','WriteMode','
            overwritesheet');
        writetable(TGNSNN,'TFP_GO_nsm_nn.xlsx','FileType','spreadsheet','Sheet','TFP_GO_nsm_nn','
            WriteMode','overwritesheet');
    otherwise
        fprintf('Incorrect variable. Set Var from any one of the following:\n RealGO \n RealII \n
            NominalVA \n NominalGO \n NominalII \n IIShare \n Employment \n LaborSharesUnScaled \n
            SelfEmployed \n ScalingFactor \n LaborShares \n NominalCapital \nDepreciationRate \n
            NominalInvestment \n RealInvestment \n RealInvestmentDollars \n TFP \n');
end
cd(Dir.Working);
if Var ~= "TFP"
    Sourcefile = strcat(Dir.Output,'/37 Sector Data.xlsx');
    Destination = Dir.DataPr;
    StatusPr   = copyfile(Sourcefile,Destination);
else
    Sourcefile = strcat(Dir.Output,'/TFP_GO.xlsx');
    Destination = Dir.DataPr;
    StatusPr1   = copyfile(Sourcefile,Destination);
    Sourcefile = strcat(Dir.Output,'/TFP_GO_sm.xlsx');
    Destination = Dir.DataPr;
    StatusPr2  = copyfile(Sourcefile,Destination);
    Sourcefile = strcat(Dir.Output,'/TFP_GO_nsm.xlsx');
    Destination = Dir.DataPr;
    StatusPr3  = copyfile(Sourcefile,Destination);
    Sourcefile = strcat(Dir.Output,'/TFP_GO_nsm_nn.xlsx');
    Destination = Dir.DataPr;
    StatusPr4  = copyfile(Sourcefile,Destination);
    StatusPr   = (StatusPr1&StatusPr2&StatusPr3&StatusPr4);
end
cd(Dir.Working);

```

```
end
```

11.7 Figures 4-8 and Supplemental Appendix Figures 1-6

11.7.1 main.m

```
Inputs:
flag.time_varying_beta: A flag variable indicating whether to use time-varying consumption allocation
    coefficients.
save_fig_flag: A flag variable indicating whether to save figures.
est_flag: A flag variable indicating whether to perform parameter optimization.
endo_lambda_flag: A flag variable indicating whether to use endogenous lambda parameter.
post_calib_flag: A flag variable indicating whether to perform post-calibration analysis.
LS_min: Minimum labor share allowed, set to 0.02.
Data loaded from load_data function, including:
alpha_data: A 3D matrix containing allocation coefficients for different periods.
tfp_data: A matrix containing total factor productivity data for different periods.
Cons47bea: A matrix containing consumption data starting from 1947.
price_va: A matrix containing price data for different periods.
VA47bea: A matrix containing value-added data starting from 1947.
vec_star: A vector containing the results of the parameter optimization.

Outputs:
output_real: A vector containing the log values of the actual output.
domar_weight_data: A matrix containing Domar weights for different periods.
cons_gr_data: A vector containing the consumption growth data.

Auxiliary Code:
Data Preprocessing: The load_data function loads the necessary datasets.
Parameter Optimization: The fmincon function is used to perform parameter optimization if est_flag is set
     to 1.
Post-Calibration Analysis: The analysis_calib and analysis_fixed_rho_calib functions are used to perform
    post-calibration analysis if post_calib_flag is set to 1.
```

```
% Clean version of the code to reproduce results

clear
clc
```

```
close all
warning('off','all')

%flags: if all flags are zero, baseline model is run
flag=struct;
flag.time_varying_beta=0;

if flag.time_varying_beta>1
   disp('Problem with flags! Pick a version of the model in main.m')
end

%this variable saves figures (if ==1)
save_fig_flag=1;

%if est_flag==1, make sure that fmincon converges
%in one step
est_flag=0;

%% preliminaries

%if endo_lambda_flag=0, use lambda=0.37 (estimate_tfp)
endo_lambda_flag=1;

%if post_calib_flag=1 then do some post-calibration analysis
post_calib_flag=1;

LS_min = 0.02; % Minimum labor share allowed (this number is irrelevant as long as it is small)

%% input: data on shares, consumption, prices
load_data;

[n,T]=size(tfp_data);

% Set the alpha_bar (ideal shares) to their data mean
alpha_bar = zeros(n,n+1);
alpha_bar(:,1:n) = mean(alpha_data,3);
alpha_bar(:,end) = sum(alpha_bar(:,1:n),2);

%% process data
```

```
%consumption: start from 1948
Cons48bea=Cons47bea(2:end,:);

if flag.time_varying_beta==0
    avg_cons_t = zeros(n,T);
    for t=1:T
        avg_cons_t(:,t) = Cons48bea(t,:)./sum(Cons48bea(t,:));
    end
    avg_cons = mean(avg_cons_t,2);
    beta = avg_cons./sum(avg_cons);

    %data-implied consumption as prescribed by the model
    Cons48bea_real=Cons48bea./price_va;
    C_data_sector=log(Cons48bea_real).*beta';
    C_data_sector(C_data_sector==-Inf)=0;
    C_data=sum(C_data_sector,2);
    cons_gr_data=C_data(2:end)-C_data(1:end-1); %already in logs

    % data Domar weight
    domar_weight_data = zeros(n,T);
    for t=1:T
        domar_weight_data(:,t) = beta'/(eye(n,n)-alpha_data(:,:,t));
    end
end

if flag.time_varying_beta==1
    beta = zeros(n,T);
    for t=1:T
        beta(:,t) = Cons48bea(t,:)./sum(Cons48bea(t,:));
    end

    %data-implied consumption as prescribed by the model
    C_data = zeros(T,1);

    for t=1:T
        Cons48bea_real_t = Cons48bea(t,:)./price_va(t,:);
        C_data_sector_t = log(Cons48bea_real_t).*beta(:,t)';
        C_data_sector_t(C_data_sector_t==-Inf)=0;
        C_data_sector_t(isnan(C_data_sector_t))=0;
```

```
        C_data(t) = sum(C_data_sector_t);
    end

    cons_gr_data=C_data(2:end)-C_data(1:end-1); %already in logs

    % data Domar weight
    domar_weight_data = zeros(n,T);
    for t=1:T
        domar_weight_data(:,t) = beta(:,t)'/(eye(n,n)-alpha_data(:,:,t));
    end
end

%output
output_real=log(sum(VA47bea(2:end,:)./price_va,2));

%%
% load vector of parameters that is the result of optimization
load vec_star

%confirm that we converge to this point if start from it
if est_flag==1
    options_optim = optimoptions(@fmincon,'MaxIterations',1000,'MaxFunctionEvaluations',1e6,'Display','
        iter','Algorithm','sqp');
    f = @(x) obj_calib_kappa_est(x,alpha_bar,alpha_data,tfp_data,LS_min,beta,endo_lambda_flag,0,flag,
        cons_gr_data);
    lb_kappa = 0.01;
    ub_kappa = 60;
    lb_rho = 1;
    ub_rho = 10;
    [vec_star_conv] = fmincon(f,vec_star,[],[],[],[],[lb_kappa*ones(2*n,1);lb_rho],[ub_kappa*ones(2*n,1);
        ub_rho],[],options_optim);
end

%% post-calibration exercises

save_folder='../../../output_figures';

if post_calib_flag==1
    analysis_calib;
```

```
    if flag.time_varying_beta==0
       save_folder='../../output_figures';
       analysis_fixed_rho_calib;
    end
end
```

11.7.2 analysis_calib.m

```
Input Variables
vec_star: Optimal parameter vector obtained from the calibration
alpha_bar: Ideal shares, mean of alpha_data
alpha_data: Data on alpha (shares)
tfp_data: Total factor productivity data
LS_min: Minimum labor share allowed
beta: Consumption shares
endo_lambda_flag: Flag for endogenous lambda usage
flag: Structure with various flags (e.g., time_varying_beta)
epsilon: Shocks
mu_drift: Drift in the shocks
sigma_t: Volatility matrix over time
a0: Initial parameter for the a_alpha_star function
rho: Risk aversion parameter
n: Number of sectors
T: Number of time periods
sector_names: Names of the sectors

Output Variables
kappa: Parameter from obj_calib_kappa
alpha_star: Calibrated alpha star matrix
C: Consumption vector
EC: Expected consumption
VC: Variance of consumption
mu_drift: Drift in the shocks (unchanged)
sigma_t: Volatility matrix over time (unchanged)
epsilon: Shocks (unchanged)
a0: Initial parameter for the a_alpha_star function (unchanged)
rho: Risk aversion parameter (unchanged)
kappa_i: Kappa parameters for sector i
```

```
kappa_j: Kappa parameters for sector j
tfp_model: TFP model
lambda: Weight of past observations in estimation of volatility
a_tfp_data: TFP data parameter
zeta_tfp_data: TFP data parameter
a_tfp_model: TFP model parameter
zeta_tfp_model: TFP model parameter
alpha_star_base: Alpha star matrix for the baseline scenario
domar_star_base: Domar weight matrix for the baseline scenario
C_base: Consumption vector for the baseline scenario
EC_base: Expected consumption for the baseline scenario
VC_base: Variance of consumption for the baseline scenario
alpha_star_sig0: Alpha star matrix for the zero sigma scenario
domar_star_sig0: Domar weight matrix for the zero sigma scenario
C_sig0: Consumption vector for the zero sigma scenario
EC_sig0: Expected consumption for the zero sigma scenario
VC_sig0: Variance of consumption for the zero sigma scenario
alpha_star_known_shocks: Alpha star matrix for the known shocks scenario
domar_star_known_shocks: Domar weight matrix for the known shocks scenario
C_known_shocks: Consumption vector for the known shocks scenario
W_base: Welfare for the baseline scenario
W_sig0: Welfare for the zero sigma scenario
alpha_star_fixed: Fixed alpha star matrix
C_fixed: Fixed consumption vector
EC_fixed: Fixed expected consumption
VC_fixed: Fixed variance of consumption
W_fixed: Fixed welfare
EC_known_shocks: Known shocks expected consumption
VC_known_shocks: Known shocks variance of consumption
W_known_shocks: Known shocks welfare

Auxiliary Code (External Function Calls)

obj_calib_kappa(vec_star, alpha_bar, alpha_data, tfp_data, LS_min, beta, endo_lambda_flag, 0, flag)

compute_eq_time_series(epsilon, mu_drift, sigma_t, kappa, alpha_bar, rho, LS_min, beta, a0, 0, flag)

bar_func(n, kappa, alpha_bar)

```

```
a_alpha_star(alpha_star_cur, n, A_bar, B_bar, C_bar, a0)

fig_kappas.m

results_TFP.m

results_trends.m

results_Sigma_matrix.m

fig_TFP.m

fig_domar.m

fig_GR.m

results_counterfactuals.m

results_domar.m

results_correlations.m
```

```
% This script should be run after an optimal vec_star has been found. In
% constructs figure and provides summary statistics about the calibrated economy



close all
clc

sector_names = {'Mining','Utilities','Construction','Wood products','Nonmetallic minerals','Primary
    metals','Fabricated metals',...
    'Machinery','Computer and electronic','Electrical equipment','Motor vehicles','Other transp. equip.','
        Furniture',...
    'Misc. manufacturing','Food and beverage','Textile','Apparel','Paper','Printing','Petroleum','Chemical
        ','Plastics',...
    'Wholesale trade','Retail trade','Transp. and warehousing','Information','Finance and insurance','Real
         estate',...
    'Prof. and tech. services','Management','Admin. services','Education','Health care','Arts','
        Accommodation',...
```

```
    'Food services','Other'};

%% benchmark stats
[kappa,alpha_bar,alpha_star,C,EC,VC,mu_drift,sigma_t,epsilon,a0,rho,kappa_i,kappa_j,tfp_model,lambda,
    a_tfp_data,zeta_tfp_data,a_tfp_model,zeta_tfp_model] = ...
    obj_calib_kappa(vec_star,alpha_bar,alpha_data,tfp_data,LS_min,beta,endo_lambda_flag,0,flag);

disp(" ");
disp(['Risk aversion parameter, rho                        = ',num2str(rho)]);
disp(['Weight of past observations in estimation of volatility, phi = ',num2str(lambda)]);
disp(" ");


%% Solve with and without uncertainty
[alpha_star_base,domar_star_base,C_base,EC_base,VC_base] = compute_eq_time_series(epsilon,mu_drift,
    sigma_t,kappa,alpha_bar,rho,LS_min,beta,a0,0,flag);
[alpha_star_sig0,domar_star_sig0,C_sig0,EC_sig0,VC_sig0] = compute_eq_time_series(epsilon,mu_drift,
    sigma_t,kappa,alpha_bar,1,LS_min,beta,a0,0,flag);
[alpha_star_known_shocks,domar_star_known_shocks,C_known_shocks] = compute_eq_time_series(epsilon,
    mu_drift,sigma_t,kappa,alpha_bar,1,LS_min,beta,a0,1,flag);

W_base = EC_base - 0.5*(rho-1)*VC_base;
W_sig0 = EC_sig0 - 0.5*(rho-1)*VC_sig0;

%% Compute the fixed network economy and known shocks
alpha_star_fixed=repmat(mean(alpha_star_base,3),1,1,T);
C_fixed = zeros(T,1);
EC_fixed = zeros(T,1);
VC_fixed = zeros(T,1);
W_fixed = zeros(T,1);
EC_known_shocks=zeros(T,1);
VC_known_shocks=zeros(T,1);
W_known_shocks=zeros(T,1);

[A_bar,B_bar,C_bar] = bar_func(n,kappa,alpha_bar);

mu_t=zeros(n,T);
mu_t(:,1)=epsilon(:,1);
mu_t(:,2:end)=epsilon(:,1:end-1)+mu_drift;
```

```
for t=1:T
   beta_cur=beta;
   if flag.time_varying_beta==1
      beta_cur=beta(:,t);
   end
   mu=mu_t(:,t);
   sigma = sigma_t(:,:,t);
   alpha_star_cur=alpha_star_fixed(:,:,t);
   inv_L = (eye(n) - alpha_star_cur);
   C_fixed(t) = beta_cur'/inv_L*(epsilon(:,t) + a_alpha_star(alpha_star_cur,n,A_bar,B_bar,C_bar,a0));
   EC_fixed(t) = beta_cur'/inv_L*(mu+a_alpha_star(alpha_star_cur,n,A_bar,B_bar,C_bar,a0));
   VC_fixed(t) = beta_cur'*(inv_L\sigma/inv_L')*beta_cur;
   W_fixed(t) = EC_fixed(t) - 0.5*(rho-1)*VC_fixed(t);


   alpha_star_cur=alpha_star_known_shocks(:,:,t);
   inv_L = (eye(n) - alpha_star_cur);
   EC_known_shocks(t) = beta_cur'/inv_L*(mu+a_alpha_star(alpha_star_cur,n,A_bar,B_bar,C_bar,a0));
   VC_known_shocks(t) = beta_cur'*(inv_L\sigma/inv_L')*beta_cur;
   W_known_shocks(t) = EC_known_shocks(t) - 0.5*(rho-1)*VC_known_shocks(t);
end


%% kappas: plots and stats
if flag.time_varying_beta==0
   run("aux_figures_codes/fig_kappas.m")
end

%% sectoral shocks; TFP in the data and in the model
if flag.time_varying_beta==0
   run("aux_figures_codes/results_TFP.m")
end


%% sectoral trends
if flag.time_varying_beta==0
   run("aux_figures_codes/results_trends.m")
end

```

```
%% time-series of TFP and uncertainty
if flag.time_varying_beta==0

    %statistics on Sigma
    run("aux_figures_codes/results_Sigma_matrix.m")

    %mu and Sigma: time series plot
    run("aux_figures_codes/fig_TFP.m")
end





%% Graph Domar weights in data and model
if flag.time_varying_beta==0
    run("aux_figures_codes/fig_domar.m")
end

%% comparison across different models: GR and full sample
run("aux_figures_codes/fig_GR.m")



%% Counterfactuals: fixed network, no uncertainty, known_shocks
if flag.time_varying_beta==0
    run("aux_figures_codes/results_counterfactuals.m")
end

%% Domar weights in the data and in the model
if flag.time_varying_beta==0
    run("aux_figures_codes/results_domar.m")
end

%% Correlations between sectos
if flag.time_varying_beta==0
    run("aux_figures_codes/results_correlations.m")
end
```

11.7.3 obj_calib_kappa.m

```
Input Variables
x: Parameter vector to be calibrated
alpha_bar: Ideal shares, mean of alpha_data
alpha_data: Data on alpha (shares)
tfp_data: Total factor productivity data
LS_min: Minimum labor share allowed
beta: Consumption shares
endo_lambda_flag: Flag for endogenous lambda usage
known_shocks_flag: Flag indicating if known shocks are considered
flag: Structure with various flags (e.g., time_varying_beta)

Output Variables
kappa: Calibrated kappa parameter matrix
alpha_bar: Ideal shares, mean of alpha_data (unchanged)
alpha_star: Calibrated alpha star matrix
C: Consumption vector
EC: Expected consumption
VC: Variance of consumption
mu_drift: Drift in the shocks
sigma_t: Volatility matrix over time
epsilon: Shocks
a0: Initial parameter for the a_alpha_star function
rho: Risk aversion parameter
kappa_i: Kappa parameters for sector i
kappa_j: Kappa parameters for sector j
TFP: Total factor productivity
lambda: Weight of past observations in estimation of volatility
a_tfp_data: TFP data parameter
zeta_tfp_data: TFP data parameter
a_tfp_model: TFP model parameter
zeta_tfp_model: TFP model parameter

Auxiliary Code (External Function Calls)
estimate_tfp(tfp_data, alpha_bar, alpha_data, kappa, a0, endo_lambda_flag)
```

```
compute_eq_time_series(epsilon, mu_drift, sigma_t, kappa, alpha_bar, rho, LS_min, beta, a0,
    known_shocks_flag, flag)
```

```
function [kappa,alpha_bar,alpha_star,C,EC,VC,mu_drift,sigma_t,epsilon,a0,rho,kappa_i,kappa_j,TFP,lambda,
    a_tfp_data,zeta_tfp_data,a_tfp_model,zeta_tfp_model] = ...
   obj_calib_kappa(x,alpha_bar,alpha_data,tfp_data,LS_min,beta,endo_lambda_flag,known_shocks_flag,flag)

[n,~] = size(tfp_data);
a0=mean(tfp_data,2);

if isrow(x)
   x =x';
end

kappa_i = [15; x(1:n-1)];
kappa_j = x(n:2*n);
kappa = kappa_i*kappa_j';
rho = x(end);


[mu_drift,sigma_t,a_tfp_data,zeta_tfp_data,epsilon,lambda] = estimate_tfp(tfp_data,alpha_bar,alpha_data,
    kappa,a0,endo_lambda_flag);
[alpha_star,~,C,EC,VC,TFP,a_tfp_model,zeta_tfp_model] = compute_eq_time_series(epsilon,mu_drift,sigma_t,
    kappa,alpha_bar,rho,LS_min,beta,a0,known_shocks_flag,flag);


end
```

11.7.4 compute_eq_time_series.m

```
Input Variables
epsilon: Shocks data (n x T matrix)
mu_drift: Drift in the shocks (n x 1 vector)
sigma_t: Volatility matrix over time (n x n x T tensor)
kappa: Calibrated kappa parameter matrix (n x n matrix)
alpha_bar: Ideal shares, mean of alpha_data (n x n+1 matrix)
rho: Risk aversion parameter (scalar)
LS_min: Minimum labor share allowed (scalar)
```

```
beta: Consumption shares (n x T or n x 1 matrix)
a0: Initial parameter for the a_alpha_star function (n x 1 vector)
known_shocks_flag: Flag indicating if known shocks are considered (scalar)
flag: Structure with various flags (e.g., time_varying_beta)

Output Variables
alpha_star: Calibrated alpha star matrix (n x n x T tensor)
domar_star: Domar weight matrix (n x T matrix)
C: Consumption vector (T x 1 vector)
EC: Expected consumption (T x 1 vector)
VC: Variance of consumption (T x 1 vector)
TFP: Total factor productivity (n x T matrix)
a_alpha_star_tfp: TFP data parameter (n x T matrix)
zeta_tfp: TFP data parameter (n x T matrix)
error_flag: Error flag for each time period (T x 1 vector)

Auxiliary Code (External Function Calls)
bar_func(n, kappa, alpha_bar)
compute_eq(kappa, alpha_bar, mu, sigma, rho, LS_min, beta, epsilon, A_bar, B_bar, C_bar, a0)
```

```
function [alpha_star,domar_star,C,EC,VC,TFP,a_alpha_star_tfp,zeta_tfp,error_flag] =
    compute_eq_time_series(epsilon,mu_drift,sigma_t,kappa,alpha_bar,rho,LS_min,beta,a0,known_shocks_flag,
    flag)

n = size(epsilon,1);
T = size(epsilon,2);

[A_bar,B_bar,C_bar] = bar_func(n,kappa,alpha_bar);

alpha_star = zeros(n,n,T); % Keep the equilibrium input shares
domar_star = zeros(n,T);

C = zeros(T,1);
EC = zeros(T,1);
VC = zeros(T,1);
error_flag = zeros(T,1);

TFP=zeros(n,T);
a_alpha_star_tfp=zeros(n,T);
```

```
zeta_tfp=zeros(n,T);
mu_t=zeros(n,T);
mu_t(:,1)=epsilon(:,1);
mu_t(:,2:end)=epsilon(:,1:end-1)+mu_drift;
if known_shocks_flag==0
    for t=1:T
        mu=mu_t(:,t);
        sigma = sigma_t(:,:,t);
        if flag.time_varying_beta==0
            [alpha_star(:,:,t),domar_star(:,t),C(t),EC(t),VC(t),TFP(:,t),error_flag(t),a_alpha_star_tfp(:,t
                ), zeta_tfp(:,t)] = compute_eq(kappa,alpha_bar,mu,sigma,rho,LS_min,beta,epsilon(:,t),A_bar,
                B_bar,C_bar,a0);
        else
            [alpha_star(:,:,t),domar_star(:,t),C(t),EC(t),VC(t),TFP(:,t),error_flag(t),a_alpha_star_tfp(:,t
                ), zeta_tfp(:,t)] = compute_eq(kappa,alpha_bar,mu,sigma,rho,LS_min,beta(:,t),epsilon(:,t),
                A_bar,B_bar,C_bar,a0);
        end
    end
else
    mu_t=epsilon;
    for t=1:T
        mu=mu_t(:,t);
        sigma = zeros(n,n);
        if flag.time_varying_beta==0
            [alpha_star(:,:,t),domar_star(:,t),C(t),EC(t),VC(t),TFP(:,t),error_flag(t),a_alpha_star_tfp(:,t
                ), zeta_tfp(:,t)] = compute_eq(kappa,alpha_bar,mu,sigma,rho,LS_min,beta,epsilon(:,t),A_bar,
                B_bar,C_bar,a0);
        else
            [alpha_star(:,:,t),domar_star(:,t),C(t),EC(t),VC(t),TFP(:,t),error_flag(t),a_alpha_star_tfp(:,t
                ), zeta_tfp(:,t)] = compute_eq(kappa,alpha_bar,mu,sigma,rho,LS_min,beta(:,t),epsilon(:,t),
                A_bar,B_bar,C_bar,a0);
        end
    end
end

end
```

11.7.5 bar_func.m

```
### Input Variables

1. `n`: Number of sectors (scalar).
2. `kappa`: Calibrated kappa parameter matrix (n x n+1 matrix).
3. `alpha_bar`: Ideal shares, mean of alpha_data (n x n+1 matrix).

### Output Variables

1. `A_bar`: Auxiliary matrix for the `a_alpha_star` function (n x n x n tensor).
2. `B_bar`: Auxiliary matrix for the `a_alpha_star` function (n x n matrix).
3. `C_bar`: Auxiliary matrix for the `a_alpha_star` function (n x 1 vector).

### Auxiliary Code (External Function Calls)

1. `fig_kappas.m`: Generates plots and statistics for kappas.
2. `results_TFP.m`: Calculates and plots TFP in the data and model.
3. `results_trends.m`: Calculates and plots sectoral trends.
4. `results_Sigma_matrix.m`: Statistics on the Sigma matrix.
5. `fig_TFP.m`: Plots time series of TFP and uncertainty.
6. `fig_domar.m`: Graphs Domar weights in data and model.
7. `fig_GR.m`: Comparison across different models (GR and full sample).
8. `results_counterfactuals.m`: Counterfactual analysis (fixed network, no uncertainty, known shocks).
9. `results_domar.m`: Calculates and plots Domar weights in the data and model.
10. `results_correlations.m`: Calculates and plots correlations between sectors.
```

```
function [A_bar,B_bar,C_bar] = bar_func(n,kappa,alpha_bar)

%compute A_bar, B_bar and C_bar: useful matrices to compute TFP penalty due to deviation from ideal
    shares



    A_bar=zeros(n,n,n);
    C_bar=zeros(n,1);

    B_bar=(kappa(:,1:n).*alpha_bar(:,1:n)+kappa(:,n+1).*alpha_bar(:,n+1))';
    for i_firm=1:n
```

```
        a_bar = -ones(n,n)*kappa(i_firm,n+1);
        a_bar(logical(eye(n))) = a_bar(logical(eye(n))) - kappa(i_firm,1:n)';
        A_bar(:,:,i_firm)=a_bar;
        C_bar(i_firm) = -(kappa(i_firm,n+1)*(alpha_bar(i_firm,n+1))^2+sum(kappa(i_firm,1:n).*alpha_bar(
            i_firm,1:n).^2));
    end

    A_bar=A_bar/2;
    C_bar=C_bar/2;
end
```

11.7.6 a_alpha_star.m

```
### Input Variables

1. `alpha_star`: Equilibrium input shares (n x n matrix).
2. `n`: Number of sectors (scalar).
3. `A_bar`: Auxiliary matrix for the `a_alpha_star` function (n x n x n tensor).
4. `B_bar`: Auxiliary matrix for the `a_alpha_star` function (n x n matrix).
5. `C_bar`: Auxiliary matrix for the `a_alpha_star` function (n x 1 vector).
6. `a0`: Mean of TFP data (n x 1 vector).

### Output Variables

1. `a_star`: Equilibrium TFP term coming from the input choice (n x 1 vector).

### Auxiliary Code (External Function Calls)

1. `fig_kappas.m`: Generates plots and statistics for kappas.
2. `results_TFP.m`: Calculates and plots TFP in the data and model.
3. `results_trends.m`: Calculates and plots sectoral trends.
4. `results_Sigma_matrix.m`: Statistics on the Sigma matrix.
5. `fig_TFP.m`: Plots time series of TFP and uncertainty.
6. `fig_domar.m`: Graphs Domar weights in data and model.
7. `fig_GR.m`: Comparison across different models (GR and full sample).
8. `results_counterfactuals.m`: Counterfactual analysis (fixed network, no uncertainty, known shocks).
9. `results_domar.m`: Calculates and plots Domar weights in the data and model.
10. `results_correlations.m`: Calculates and plots correlations between sectors.
```

```
function [a_star] = a_alpha_star(alpha_star,n,A_bar,B_bar,C_bar,a0)
% Compute a(alpha_star), the equilibrium TFP term coming from the input choice

t1=(sum(alpha_star'.*B_bar))';
t2=pagemtimes(reshape(alpha_star',1,n,n),A_bar);
t2=squeeze(pagemtimes(t2,reshape(alpha_star',n,1,n)));
a_star=a0+t1+t2+C_bar;



end
```

11.7.7 analysis_fixed_rho_calib.m

```
### Input Variables

1. `vec_star`: Optimized parameter vector.
2. `tfp_data`: Total Factor Productivity data.
3. `alpha_bar`: Ideal shares.
4. `alpha_data`: Actual shares data.
5. `kappa`: Parameter matrix.
6. `a0`: Mean of TFP data.
7. `endo_lambda_flag`: Flag for endogenous lambda.
8. `flag`: Struct containing model flags.
9. `LS_min`: Minimum labor share allowed.
10. `beta`: Consumption shares.
11. `save_fig_flag`: Flag to save figures.
12. `save_folder`: Folder to save figures.
13. `T`: Number of time periods.
14. `rho_arr`: Array of risk aversion parameters.
15. `rho_init`: Initial risk aversion parameter.
16. `N_rho`: Number of risk aversion parameters.

### Output Variables

1. `C_base_arr`: Baseline consumption array (T x N_rho).
2. `EC_base_arr`: Baseline expected consumption array (T x N_rho).
3. `VC_base_arr`: Baseline consumption variance array (T x N_rho).
```

```
4. `W_base_arr`: Baseline welfare array (T x N_rho).
5. `C_sig0_arr`: Consumption array with sigma=0 (T x N_rho).
6. `EC_sig0_arr`: Expected consumption array with sigma=0 (T x N_rho).
7. `VC_sig0_arr`: Consumption variance array with sigma=0 (T x N_rho).
8. `W_sig0_arr`: Welfare array with sigma=0 (T x N_rho).

### Auxiliary Code (External Function Calls)

1. `fig_kappas.m`: Generates plots and statistics for kappas.
2. `results_TFP.m`: Calculates and plots TFP in the data and model.
3. `results_trends.m`: Calculates and plots sectoral trends.
4. `results_Sigma_matrix.m`: Statistics on the Sigma matrix.
5. `fig_TFP.m`: Plots time series of TFP and uncertainty.
6. `fig_domar.m`: Graphs Domar weights in data and model.
7. `fig_GR.m`: Comparison across different models (GR and full sample).
8. `results_counterfactuals.m`: Counterfactual analysis (fixed network, no uncertainty, known shocks).
9. `results_domar.m`: Calculates and plots Domar weights in the data and model.
10. `results_correlations.m`: Calculates and plots correlations between sectors.
11. `estimate_tfp.m`: Estimates TFP, mu, sigma, and epsilon.
12. `compute_eq_time_series.m`: Computes equilibrium time series.
13. `compute_eq.m`: Computes equilibrium.
14. `bar_func.m`: Computes auxiliary matrices A_bar, B_bar, and C_bar.
15. `a_alpha_star.m`: Computes equilibrium TFP term from input choice.
```

```

rho_arr=[vec_star(end), 2, 10];
N_rho=length(rho_arr);

rho_init=vec_star(end);

C_base_arr=zeros(T,N_rho);
EC_base_arr=zeros(T,N_rho);
VC_base_arr=zeros(T,N_rho);
W_base_arr=zeros(T,N_rho);
C_sig0_arr=zeros(T,N_rho);
EC_sig0_arr=zeros(T,N_rho);
VC_sig0_arr=zeros(T,N_rho);
W_sig0_arr=zeros(T,N_rho);

```

```
LegendsStrings = cell(N_rho,1); % Initialize array with legends

for i_rho=1:N_rho

    rho=rho_arr(i_rho);

    % Compute the implies mu, sigma and epsilon
    [mu_drift,sigma_t,a_tfp,zeta_tfp,epsilon] = estimate_tfp(tfp_data,alpha_bar,alpha_data,kappa,a0,
        endo_lambda_flag);

    [alpha_star_base,domar_star_base,C_base,EC_base,VC_base] = compute_eq_time_series(epsilon,mu_drift,
        sigma_t,kappa,alpha_bar,rho,LS_min,beta,a0,0,flag);
    [alpha_star_sig0,domar_star_sig0,C_sig0,EC_sig0,VC_sig0] = compute_eq_time_series(epsilon,mu_drift,
        sigma_t,kappa,alpha_bar,1,LS_min,beta,a0,0,flag);

    C_base_arr(:,i_rho)=C_base;
    EC_base_arr(:,i_rho)=EC_base;
    VC_base_arr(:,i_rho)=VC_base;
    C_sig0_arr(:,i_rho)=C_sig0;
    EC_sig0_arr(:,i_rho)=EC_sig0;
    VC_sig0_arr(:,i_rho)=VC_sig0;

    W_base_arr(:,i_rho) = EC_base - 0.5*(rho-1)*VC_base;
    W_sig0_arr(:,i_rho) = EC_sig0 - 0.5*(rho-1)*VC_sig0;
    LegendsStrings{i_rho} = ['$$\rho = $$',num2str(rho,'%1.2f')];
end




rho=rho_init;

%% Zoom on the Great Recession
years_gr = 2006:2012;
t_start=59;
t_end=t_start+length(years_gr)-1;
years_ix = t_start:t_end;


```

```
fig = figure('Position',[100,100,800,200]);
box on
grid on
hold on
h1=plot(years_gr,100*[EC_base_arr(years_ix,1)-EC_sig0_arr(years_ix,1)],'LineWidth',2);
h2=plot(years_gr,100*[EC_base_arr(years_ix,2)-EC_sig0_arr(years_ix,2)],'--','LineWidth',2);
h3=plot(years_gr,100*[EC_base_arr(years_ix,3)-EC_sig0_arr(years_ix,3)],'-x','LineWidth',2,'markersize',10);
% h2=plot(years_gr,100*[EC_base(years_ix)-EC_fixed_GR],'--','LineWidth',2);
plot(years_gr,zeros(size(years_gr)),':k','LineWidth',1)
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)
set(gca,'TickLabelInterpreter','latex')
ylabel('E$(y) - $E$(\tilde{y})$','Interpreter','latex')
ylim([-1 0.1])
yticks([-2:0.5:0.1])
% yticklabels({'-0.25','0.0'})
legend(LegendsStrings,'Interpreter','latex','Location','southwest')
set(gcf, 'Color', 'w');
if save_fig_flag==1
   exportgraphics(gca,strcat(save_folder,'/supp_fig5/EC_GR_no_unc_rho.eps'))
   exportgraphics(gca,strcat(save_folder,'/supp_fig5/EC_GR_no_unc_rho.png'))
end


fig = figure('Position',[0,0,800,200]);
box on
grid on
hold on
h1=plot(years_gr,100*[W_base_arr(years_ix,1)-W_sig0_arr(years_ix,1)],'LineWidth',2);
h2=plot(years_gr,100*[W_base_arr(years_ix,2)-W_sig0_arr(years_ix,2)],'--','LineWidth',2);
h3=plot(years_gr,100*[W_base_arr(years_ix,3)-W_sig0_arr(years_ix,3)],'-x','LineWidth',2,'markersize',10);
% h2=plot(years_gr,100*[W_base(years_ix)-W_fixed_GR],'--','LineWidth',2);
plot(years_gr,zeros(size(years_gr)),':k','LineWidth',1)
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)
set(gca,'TickLabelInterpreter','latex')
ylabel('$\mathcal{W} - \tilde{\mathcal{W}}$','Interpreter','latex')
ylim([-0.5 2.5])
```

```
yticks([0.0:1:4])
% yticklabels({'0.0','0.0025'})
set(gcf, 'Color', 'w');
if save_fig_flag==1
   exportgraphics(gca,strcat(save_folder,'/supp_fig5/W_GR_no_unc_rho.eps'))
   exportgraphics(gca,strcat(save_folder,'/supp_fig5/W_GR_no_unc_rho.png'))
end


fig = figure('Position',[0,0,800,200]);
box on
grid on
hold on
h1=plot(years_gr,100*[sqrt(VC_base_arr(years_ix,1))-sqrt(VC_sig0_arr(years_ix,1))],'LineWidth',2);
h2=plot(years_gr,100*[sqrt(VC_base_arr(years_ix,2))-sqrt(VC_sig0_arr(years_ix,2))],'--','LineWidth',2);
h3=plot(years_gr,100*[sqrt(VC_base_arr(years_ix,3))-sqrt(VC_sig0_arr(years_ix,3))],'-x','LineWidth',2,'
    markersize',10);
% h2=plot(years_gr,100*[sqrt(VC_base(years_ix))-sqrt(VC_fixed_GR)],'--','LineWidth',2);
plot(years_gr,zeros(size(years_gr)),':k','LineWidth',1)
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)
set(gca,'TickLabelInterpreter','latex')
ylabel('$\sqrt{V(y)} - \sqrt{V(\tilde{y})}$','Interpreter','latex')
ylim([-6 0.5])
yticks([ -10:2:0])
% yticklabels({'-0.01','0.0'})
set(gcf, 'Color', 'w');
if save_fig_flag==1
   exportgraphics(gca,strcat(save_folder,'/supp_fig5/VC_GR_no_unc_rho.eps'))
   exportgraphics(gca,strcat(save_folder,'/supp_fig5/VC_GR_no_unc_rho.png'))
end

fig = figure('Position',[0,0,800,200]);
box on
grid on
hold on
h1=plot(years_gr,100*[C_base_arr(years_ix,1)-C_sig0_arr(years_ix,1)],'LineWidth',2);
h2=plot(years_gr,100*[C_base_arr(years_ix,2)-C_sig0_arr(years_ix,2)],'--','LineWidth',2);
h3=plot(years_gr,100*[C_base_arr(years_ix,3)-C_sig0_arr(years_ix,3)],'-x','LineWidth',2,'markersize',10);
```

```
% h2=plot(years_gr,100*[C_base(years_ix)-C_fixed_GR],'--','LineWidth',2);
plot(years_gr,zeros(size(years_gr)),':k','LineWidth',1)
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)
set(gca,'TickLabelInterpreter','latex')
ylabel('$y - \tilde{y}$','Interpreter','latex')
ylim([-1 6])
yticks([-0:2:10])
set(gcf, 'Color', 'w');
if save_fig_flag==1
    exportgraphics(gca,strcat(save_folder,'/supp_fig5/C_GR_no_unc_rho.eps'))
    exportgraphics(gca,strcat(save_folder,'/supp_fig5/C_GR_no_unc_rho.png'))
end



%% Display some statistics
disp(" ");
disp("SUPPLEMENTAL APPENDIX: TABLE 2");
for i_rho=1:N_rho
    disp(['Risk aversion = ', num2str(rho_arr(i_rho))]);
    disp(['mean W_base - mean W_sig0 = ',num2str(100*mean(W_base_arr(:,i_rho)-W_sig0_arr(:,i_rho)))]);
    disp(['mean EC_base - mean EC_sig0 = ',num2str(100*mean(EC_base_arr(:,i_rho)-EC_sig0_arr(:,i_rho)))]);
    disp(['mean sqrt VC_base - sqrt VC_sig0 = ',num2str(100*mean(sqrt(VC_base_arr(:,i_rho))-sqrt(
        VC_sig0_arr(:,i_rho))))]);
end
```

11.7.8 compute_eq.m

```
### Input Variables

1. `vec_star`: Optimized parameter vector.
2. `tfp_data`: Total Factor Productivity data.
3. `alpha_bar`: Ideal shares.
4. `alpha_data`: Actual shares data.
5. `kappa`: Parameter matrix.
6. `a0`: Mean of TFP data.
7. `endo_lambda_flag`: Flag for endogenous lambda.
```

```
8. `flag`: Struct containing model flags.
9. `LS_min`: Minimum labor share allowed.
10. `beta`: Consumption shares.
11. `save_fig_flag`: Flag to save figures.
12. `save_folder`: Folder to save figures.
13. `T`: Number of time periods.
14. `rho_arr`: Array of risk aversion parameters.
15. `rho_init`: Initial risk aversion parameter.
16. `N_rho`: Number of risk aversion parameters.
17. `mu`: Drift term of TFP.
18. `sigma`: Variance term of TFP.
19. `epsilon`: Shock term of TFP.
20. `A_bar`: Auxiliary matrix A_bar.
21. `B_bar`: Auxiliary matrix B_bar.
22. `C_bar`: Auxiliary matrix C_bar.

### Output Variables

1. `C_base_arr`: Baseline consumption array (T x N_rho).
2. `EC_base_arr`: Baseline expected consumption array (T x N_rho).
3. `VC_base_arr`: Baseline consumption variance array (T x N_rho).
4. `W_base_arr`: Baseline welfare array (T x N_rho).
5. `C_sig0_arr`: Consumption array with sigma=0 (T x N_rho).
6. `EC_sig0_arr`: Expected consumption array with sigma=0 (T x N_rho).
7. `VC_sig0_arr`: Consumption variance array with sigma=0 (T x N_rho).
8. `W_sig0_arr`: Welfare array with sigma=0 (T x N_rho).
9. `alpha_star`: Equilibrium input shares.
10. `domar`: Domar weights.
11. `C`: Consumption.
12. `EC`: Expected consumption.
13. `VC`: Consumption variance.
14. `TFP`: Total Factor Productivity.
15. `error_flag`: Error flag for convergence.
16. `a_alpha_star_temp`: Equilibrium TFP term.
17. `zeta_tfp`: Adjustment factor for TFP.

### Auxiliary Code (External Function Calls)

1. `fig_kappas.m`: Generates plots and statistics for kappas.
```

```
2. `results_TFP.m`: Calculates and plots TFP in the data and model.
3. `results_trends.m`: Calculates and plots sectoral trends.
4. `results_Sigma_matrix.m`: Statistics on the Sigma matrix.
5. `fig_TFP.m`: Plots time series of TFP and uncertainty.
6. `fig_domar.m`: Graphs Domar weights in data and model.
7. `fig_GR.m`: Comparison across different models (GR and full sample).
8. `results_counterfactuals.m`: Counterfactual analysis (fixed network, no uncertainty, known shocks).
9. `results_domar.m`: Calculates and plots Domar weights in the data and model.
10. `results_correlations.m`: Calculates and plots correlations between sectors.
11. `estimate_tfp.m`: Estimates TFP, mu, sigma, and epsilon.
12. `compute_eq_time_series.m`: Computes equilibrium time series.
13. `compute_eq.m`: Computes equilibrium.
14. `bar_func.m`: Computes auxiliary matrices A_bar, B_bar, and C_bar.
15. `a_alpha_star.m`: Computes equilibrium TFP term from input choice.
16. `compute_eq_firm_iteration.m`: Iterates on the firm's problem to compute equilibrium input shares.
```

```
function [alpha_star,domar,C,EC,VC,TFP,error_flag, a_alpha_star_temp, zeta_tfp] = compute_eq(kappa,
    alpha_bar,mu,sigma,rho,LS_min,beta,epsilon,A_bar,B_bar,C_bar,a0)
% Compute the equilibrium by iterating on the firm's problem.

n = size(kappa,1);

[alpha_star,error_flag] = compute_eq_firm_iteration(mu,sigma,n,rho,LS_min,beta,A_bar,B_bar,C_bar,a0,
    alpha_bar(:,1:n));



inv_L = (eye(n) - alpha_star);
domar = beta'/inv_L;
a_alpha_star_temp = a_alpha_star(alpha_star,n,A_bar,B_bar,C_bar,a0);

C = domar*(epsilon + a_alpha_star_temp);
EC = domar*(mu + a_alpha_star_temp);
VC = beta'*(inv_L\sigma/inv_L')*beta;
zeta_tfp = -log(((1-sum(alpha_star,2)).^(1-sum(alpha_star,2))).*prod(alpha_star.^alpha_star,2));
TFP=epsilon + a_alpha_star_temp + zeta_tfp;

end
```

11.7.9 compute_eq_firm_iteration.m

```
Input Variables
mu: Drift term of TFP.
sigma: Variance term of TFP.
n: Number of firms or sectors.
rho: Risk aversion parameter.
LS_min: Minimum labor share allowed.
beta: Consumption shares.
A_bar: Auxiliary matrix A_bar.
B_bar: Auxiliary matrix B_bar.
C_bar: Auxiliary matrix C_bar.
a0: Mean of TFP data.
init_alpha: Initial input shares.

Output Variables
alpha_star: Equilibrium input shares.
error_flag: Error flag for convergence.

Auxiliary Code (External Function Calls)
a_alpha_star.m: Computes equilibrium TFP term from input choice.
solve_firm_problem.m: Solves the firm's problem to compute equilibrium input shares.
```

```
function [alpha_star,error_flag] = compute_eq_firm_iteration(mu,sigma,n,rho,LS_min,beta,A_bar,B_bar,C_bar
    ,a0,init_alpha)

% This version was updated by MTD on June 21th 2021 to have a dynamically
% adapting weight on old draw in the iterations.

% Compute the equilibrium

% Initial expected benefit of each firm's input shares
% First index firm, second index input
alpha_star = init_alpha;
options_quad = optimoptions('quadprog','Display','off');

has_converged = false;
iter = 0;
iter_max = 200;
```

```
tol = 1e-4;

error_flag = false;

% Frequency of adjustment in updating weight
weight_old = 0.0;

while has_converged == false && iter<iter_max
    iter = iter+1;

    alpha_star_new = zeros(n,n);
    exitflag = zeros(n,1);

    % Compute the equilibrium TFP of the firms
    a_star = a_alpha_star(alpha_star,n,A_bar,B_bar,C_bar,a0);

    for i_firm=1:n
        A_bar_cur=A_bar(:,:,i_firm);
        B_bar_cur=B_bar(:,i_firm);
        [alpha_star_new(i_firm,:),exitflag(i_firm)] = solve_firm_problem(mu,sigma,n,rho,LS_min,beta,i_firm,
            alpha_star,a_star,A_bar_cur,B_bar_cur,options_quad);
    end

    max_diff = max(abs(alpha_star_new-alpha_star),[],'all');


    if (max_diff > 1) || (min(alpha_star_new,[],'all') < -1e-4) || (max(alpha_star_new,[],'all') > 1.0+1e
        -4) || any(isnan(alpha_star_new),'all')
        disp(alpha_star)
        disp(alpha_star_new)
        disp(exitflag)

        error("There are some issues in compute_eq_firm_iteration")
    end

    if max_diff < tol
        has_converged = true;
    else
        alpha_star = (1-weight_old).*alpha_star_new + weight_old.*alpha_star;
```

```
    end
end

if iter>=iter_max
    alpha_star = alpha_star_new;
    error_flag = true;
end

end
```

11.7.10 compute_eq_time_series.m

```
Input Variables
vec_star: Optimized parameter vector.
tfp_data: Total Factor Productivity data.
alpha_bar: Ideal shares.
alpha_data: Actual shares data.
kappa: Parameter matrix.
a0: Mean of TFP data.
endo_lambda_flag: Flag for endogenous lambda.
flag: Struct containing model flags.
LS_min: Minimum labor share allowed.
beta: Consumption shares.
save_fig_flag: Flag to save figures.
save_folder: Folder to save figures.
T: Number of time periods.
rho_arr: Array of risk aversion parameters.
rho_init: Initial risk aversion parameter.
N_rho: Number of risk aversion parameters.
mu: Drift term of TFP.
sigma: Variance term of TFP.
epsilon: Shock term of TFP.
A_bar: Auxiliary matrix A_bar.
B_bar: Auxiliary matrix B_bar.
C_bar: Auxiliary matrix C_bar.

Output Variables
C_base_arr: Baseline consumption array (T x N_rho).
```

```
EC_base_arr: Baseline expected consumption array (T x N_rho).
VC_base_arr: Baseline consumption variance array (T x N_rho).
W_base_arr: Baseline welfare array (T x N_rho).
C_sig0_arr: Consumption array with sigma=0 (T x N_rho).
EC_sig0_arr: Expected consumption array with sigma=0 (T x N_rho).
VC_sig0_arr: Consumption variance array with sigma=0 (T x N_rho).
W_sig0_arr: Welfare array with sigma=0 (T x N_rho).
alpha_star: Equilibrium input shares.
domar: Domar weights.
C: Consumption.
EC: Expected consumption.
VC: Consumption variance.
TFP: Total Factor Productivity.
error_flag: Error flag for convergence.
a_alpha_star_temp: Equilibrium TFP term.
zeta_tfp: Adjustment factor for TFP.

Auxiliary Code (External Function Calls)
fig_kappas.m: Generates plots and statistics for kappas.
results_TFP.m: Calculates and plots TFP in the data and model.
results_trends.m: Calculates and plots sectoral trends.
results_Sigma_matrix.m: Statistics on the Sigma matrix.
fig_TFP.m: Plots time series of TFP and uncertainty.
fig_domar.m: Graphs Domar weights in data and model.
fig_GR.m: Comparison across different models (GR and full sample).
results_counterfactuals.m: Counterfactual analysis (fixed network, no uncertainty, known shocks).
results_domar.m: Calculates and plots Domar weights in the data and model.
results_correlations.m: Calculates and plots correlations between sectors.
estimate_tfp.m: Estimates TFP, mu, sigma, and epsilon.
compute_eq_time_series.m: Computes equilibrium time series.
compute_eq.m: Computes equilibrium.
bar_func.m: Computes auxiliary matrices A_bar, B_bar, and C_bar.
a_alpha_star.m: Computes equilibrium TFP term from input choice.
compute_eq_firm_iteration.m: Iterates on the firm's problem to compute equilibrium input shares.
solve_firm_problem.m: Solves the firm's optimization problem.
```

```
function [alpha_star,domar_star,C,EC,VC,TFP,a_alpha_star_tfp,zeta_tfp,error_flag] =
    compute_eq_time_series(epsilon,mu_drift,sigma_t,kappa,alpha_bar,rho,LS_min,beta,a0,known_shocks_flag,
    flag)
```

```

n = size(epsilon,1);
T = size(epsilon,2);

[A_bar,B_bar,C_bar] = bar_func(n,kappa,alpha_bar);

alpha_star = zeros(n,n,T); % Keep the equilibrium input shares
domar_star = zeros(n,T);

C = zeros(T,1);
EC = zeros(T,1);
VC = zeros(T,1);
error_flag = zeros(T,1);

TFP=zeros(n,T);
a_alpha_star_tfp=zeros(n,T);
zeta_tfp=zeros(n,T);
mu_t=zeros(n,T);
mu_t(:,1)=epsilon(:,1);
mu_t(:,2:end)=epsilon(:,1:end-1)+mu_drift;
if known_shocks_flag==0
   for t=1:T
      mu=mu_t(:,t);
      sigma = sigma_t(:,:,t);
      if flag.time_varying_beta==0
         [alpha_star(:,:,t),domar_star(:,t),C(t),EC(t),VC(t),TFP(:,t),error_flag(t),a_alpha_star_tfp(:,t), zeta_tfp(:,t)] = compute_eq(kappa,alpha_bar,mu,sigma,rho,LS_min,beta,epsilon(:,t),A_bar,B_bar,C_bar,a0);
      else
         [alpha_star(:,:,t),domar_star(:,t),C(t),EC(t),VC(t),TFP(:,t),error_flag(t),a_alpha_star_tfp(:,t), zeta_tfp(:,t)] = compute_eq(kappa,alpha_bar,mu,sigma,rho,LS_min,beta(:,t),epsilon(:,t),A_bar,B_bar,C_bar,a0);
      end
   end
else
   mu_t=epsilon;
   for t=1:T
      mu=mu_t(:,t);
      sigma = zeros(n,n);
```

```
        if flag.time_varying_beta==0
            [alpha_star(:,:,t),domar_star(:,t),C(t),EC(t),VC(t),TFP(:,t),error_flag(t),a_alpha_star_tfp(:,t
                ), zeta_tfp(:,t)] = compute_eq(kappa,alpha_bar,mu,sigma,rho,LS_min,beta,epsilon(:,t),A_bar,
                B_bar,C_bar,a0);
        else
            [alpha_star(:,:,t),domar_star(:,t),C(t),EC(t),VC(t),TFP(:,t),error_flag(t),a_alpha_star_tfp(:,t
                ), zeta_tfp(:,t)] = compute_eq(kappa,alpha_bar,mu,sigma,rho,LS_min,beta(:,t),epsilon(:,t),
                A_bar,B_bar,C_bar,a0);
        end
    end
end

end
```

11.7.11 compute_eq_time_series_alt_eps.m

```
Input Variables
epsilon: Shock term of TFP (n x T matrix).
mu_drift: Drift term of TFP (n x 1 vector).
sigma_t: Time-varying covariance matrix of TFP (n x n x T array).
kappa: Parameter matrix (n x (n+1) matrix).
alpha_bar: Ideal shares (n x (n+1) matrix).
rho: Risk aversion parameter (scalar).
LS_min: Minimum labor share allowed (scalar).
beta: Consumption shares (n x 1 vector or n x T matrix).
a0: Mean of TFP data (n x 1 vector).
known_shocks_flag: Flag for known shocks (boolean).
flag: Struct containing model flags (struct).

Output Variables
alpha_star: Equilibrium input shares (n x n x T array).
domar_star: Domar weights (n x T matrix).
C: Consumption (T x 1 vector).
EC: Expected consumption (T x 1 vector).
VC: Consumption variance (T x 1 vector).
TFP: Total Factor Productivity (n x T matrix).
a_alpha_star_tfp: Equilibrium TFP term (n x T matrix).
zeta_tfp: Adjustment factor for TFP (n x T matrix).
```

```
error_flag: Error flag for convergence (T x 1 vector).

Auxiliary Code (External Function Calls)
fig_kappas.m: Generates plots and statistics for kappas.
results_TFP.m: Calculates and plots TFP in the data and model.
results_trends.m: Calculates and plots sectoral trends.
results_Sigma_matrix.m: Statistics on the Sigma matrix.
fig_TFP.m: Plots time series of TFP and uncertainty.
fig_domar.m: Graphs Domar weights in data and model.
fig_GR.m: Comparison across different models (GR and full sample).
results_counterfactuals.m: Counterfactual analysis (fixed network, no uncertainty, known shocks).
results_domar.m: Calculates and plots Domar weights in the data and model.
results_correlations.m: Calculates and plots correlations between sectors.
estimate_tfp.m: Estimates TFP, mu, sigma, and epsilon.
compute_eq_time_series.m: Computes equilibrium time series.
compute_eq.m: Computes equilibrium.
bar_func.m: Computes auxiliary matrices A_bar, B_bar, and C_bar.
a_alpha_star.m: Computes equilibrium TFP term from input choice.
compute_eq_firm_iteration.m: Iterates on the firm's problem to compute equilibrium input shares.
solve_firm_problem.m: Solves the firm's optimization problem.
```

```
function [alpha_star,domar_star,C,EC,VC,TFP,a_alpha_star_tfp,zeta_tfp,error_flag] =
    compute_eq_time_series_alt_eps(epsilon,mu_drift,sigma_t,kappa,alpha_bar,rho,LS_min,beta,a0)

n = size(epsilon,1);
T = size(epsilon,2);

[A_bar,B_bar,C_bar] = bar_func(n,kappa,alpha_bar);

alpha_star = zeros(n,n,T); % Keep the equilibrium input shares
domar_star = zeros(n,T);

C = zeros(T,1);
EC = zeros(T,1);
VC = zeros(T,1);
error_flag = zeros(T,1);

TFP=zeros(n,T);
a_alpha_star_tfp=zeros(n,T);
```

```
zeta_tfp=zeros(n,T);
mu_t=zeros(n,T);
mu_t(:,1)=epsilon(:,1);
mu_t(:,2:end)=epsilon(:,1:end-1)+mu_drift;
for t=1:T
    mu=mu_t(:,t);
    sigma = sigma_t(:,:,t);
    epsilon_cur=epsilon(:,t);


    n = size(kappa,1);

    [alpha_star(:,:,t),error_flag(t)] = compute_eq_firm_iteration(mu+1/2*diag(sigma),sigma,n,rho,LS_min,
        beta,A_bar,B_bar,C_bar,a0,alpha_bar(:,1:n));



    inv_L = (eye(n) - alpha_star(:,:,t));
    domar_star_cur = beta'/inv_L;
    a_alpha_star_tfp(:,t)= a_alpha_star(alpha_star(:,:,t),n,A_bar,B_bar,C_bar,a0);

    C(t) = domar_star_cur*(epsilon_cur + a_alpha_star_tfp(:,t));
    EC(t) = domar_star_cur*(mu + a_alpha_star_tfp(:,t));
    VC(t) = beta'*(inv_L\sigma/inv_L')*beta;
    zeta_tfp(:,t) = -log(((1-sum(alpha_star(:,:,t),2)).^(1-sum(alpha_star(:,:,t),2))).*prod(alpha_star
        (:,:,t).^alpha_star(:,:,t),2));
    TFP(:,t)=epsilon_cur + a_alpha_star_tfp(:,t) + zeta_tfp(:,t);
    domar_star(:,t)=domar_star_cur';

end


end
```

11.7.12 estimate_tfp.m

```
Input Variables
tfp_data: Total Factor Productivity data (n x T matrix).
```

```
alpha_bar: Ideal shares (n x (n+1) matrix).
alpha_data: Data-implied shares (n x n x T array).
kappa: Parameter matrix (n x (n+1) matrix).
a0: Mean of TFP data (n x 1 vector).
endo_lambda_flag: Flag for endogenous lambda (boolean).

Output Variables
mu_drift: Drift term of TFP (n x 1 vector).
sigma_t: Time-varying covariance matrix of TFP (n x n x T array).
a_tfp: Endogenous part of TFP due to shares adjustment (n x T matrix).
zeta_tfp: Adjustment factor for TFP (n x T matrix).
exo_tfp: Exogenous part of TFP as residual (n x T matrix).
lambda: Weight for time-varying variance (scalar).

Auxiliary Code (External Function Calls)
bar_func.m: Computes auxiliary matrices A_bar, B_bar, and C_bar.
a_alpha_star.m: Computes equilibrium TFP term from input choice.
garch_tfp.m: Estimates GARCH coefficients for TFP data.
```

```
function [mu_drift,sigma_t,a_tfp,zeta_tfp,exo_tfp,lambda] = estimate_tfp(tfp_data,alpha_bar,alpha_data,
    kappa,a0,endo_lambda_flag)
% This function estimate mu_drift and sigma_t from the tfp data

n = size(tfp_data,1);
T = size(tfp_data,2);

% Clean the data tfp to estimate the drift and variance
% Compute the a_tfp term
[A_bar,B_bar,C_bar] = bar_func(n,kappa,alpha_bar);

exo_tfp = zeros(n,T);
a_tfp = zeros(n,T);
zeta_tfp = zeros(n,T);

for t=1:T
    %use data-implied shares to compute zeta (normalization constant)
    zeta_tfp(:,t) = -log(((1-sum(alpha_data(:,:,t),2)).^(1-sum(alpha_data(:,:,t),2))).*prod(alpha_data
        (:,:,t).^alpha_data(:,:,t),2));
    %endogenous part due to shares adjustment
```

```
    a_tfp(:,t) = a_alpha_star(alpha_data(:,:,t),n,A_bar,B_bar,C_bar,a0);
    %get exogenous part as residual (note that everything is in logs)
    exo_tfp(:,t) = tfp_data(:,t) - a_tfp(:,t) - zeta_tfp(:,t);
end


d_tfp = exo_tfp(:,2:end)-exo_tfp(:,1:end-1);
mu_drift = mean(d_tfp,2);

sigma_t = zeros(n,n,T);


% win=11;
% lambda=0.33;
win=21;
lambda=0.37;
if endo_lambda_flag==1
    garch_coeff_avg = garch_tfp(d_tfp);
    lambda = garch_coeff_avg;
end

win_min=1;
for t=(win_min+1):T
    range = max(t-1-(win-1),1):(t-1);
    weight=lambda.^(0:(length(range)-1));
    weight=weight/sum(weight);
    weight=weight(end:-1:1);
    for j=1:n
        sigma_t(:,j,t)=sum((d_tfp(:,range)-mu_drift).*(d_tfp(j,range)-mu_drift(j)).*reshape(weight,1,
            length(range)),2);
    end
end
for t=1:win_min
    sigma_t(:,:,t) = sigma_t(:,:,win_min+1);
end



end
```

11.7.13 garch_tfp.m

```
Input Variables
tfp_data: Total Factor Productivity data (n x T matrix).
alpha_bar: Ideal shares (n x (n+1) matrix).
alpha_data: Data-implied shares (n x n x T array).
kappa: Parameter matrix (n x (n+1) matrix).
a0: Mean of TFP data (n x 1 vector).
endo_lambda_flag: Flag for endogenous lambda (boolean).

Output Variables
mu_drift: Drift term of TFP (n x 1 vector).
sigma_t: Time-varying covariance matrix of TFP (n x n x T array).
a_tfp: Endogenous part of TFP due to shares adjustment (n x T matrix).
zeta_tfp: Adjustment factor for TFP (n x T matrix).
exo_tfp: Exogenous part of TFP as residual (n x T matrix).
lambda: Weight for time-varying variance (scalar).

Auxiliary Code (External Function Calls)
bar_func.m: Computes auxiliary matrices A_bar, B_bar, and C_bar.
a_alpha_star.m: Computes equilibrium TFP term from input choice.
garch_tfp.m: Estimates GARCH coefficients for TFP data.
```

```
function garch_coeff_avg = garch_tfp(tfp)
% This function evaluates a garch(1,1) on each sectoral TFP and returns the
% average garch coefficient, i.e. how fast uncertainty decays.

n = size(tfp,1);

Mdl = garch(1,1);

garch_coeff = zeros(n,1);

for i=1:n
    EstMdl = estimate(Mdl,tfp(i,:)','Display','off');
    garch_coeff(i) = EstMdl.GARCH{1};
```

```
end

garch_coeff_avg = mean(garch_coeff);

end
```

11.7.14 hessian_func.m

```
Input Variables
tfp_data: Total Factor Productivity data (n x T matrix).
alpha_bar: Ideal shares (n x (n+1) matrix).
alpha_data: Data-implied shares (n x n x T array).
kappa: Parameter matrix (n x (n+1) matrix).
a0: Mean of TFP data (n x 1 vector).
endo_lambda_flag: Flag for endogenous lambda (boolean).

Output Variables
mu_drift: Drift term of TFP (n x 1 vector).
sigma_t: Time-varying covariance matrix of TFP (n x n x T array).
a_tfp: Endogenous part of TFP due to shares adjustment (n x T matrix).
zeta_tfp: Adjustment factor for TFP (n x T matrix).
exo_tfp: Exogenous part of TFP as residual (n x T matrix).
lambda: Weight for time-varying variance (scalar).

Auxiliary Code (External Function Calls)
bar_func.m: Computes auxiliary matrices A_bar, B_bar, and C_bar.
a_alpha_star.m: Computes equilibrium TFP term from input choice.
garch_tfp.m: Estimates GARCH coefficients for TFP data.
```

```
function hess = hessian_func(alpha,~,n,rho,beta,mu,sigma,A_bar,B_bar,C_bar)
   % hessian of the planner's problem



   alpha=reshape(alpha,n,n)';

   inv_L = eye(n)-alpha;

   %aux matrices
```

```
    v=beta'/inv_L;
    D=inv_L\sigma/inv_L';
    a = squeeze(pagemtimes(pagemtimes(reshape(alpha',1,n,n),A_bar),reshape(alpha',n,1,n))+pagemtimes(reshape(alpha',1,n,n),reshape(B_bar,n,1,n)))+C_bar;
    term0=(squeeze(2*pagemtimes(reshape(alpha',1,n,n),A_bar))+B_bar+inv_L\(mu+a))'+(1-rho)*beta'/inv_L*sigma/inv_L';


    %this version as in the notes (derivations_add.lyx) but in matrix form

%    aux1=(1-rho)*reshape(v,n,1).*reshape(v,1,1,n).*reshape(D',1,n,1,n);
%    aux2=reshape(inv(inv_L),1,n,n,1).*reshape(v,n,1).*reshape(term0,1,1,n,n);
%    aux3=permute(aux2,[3 4 1 2]);
%    aux4=2*reshape(v,n,1).*reshape(permute(param.A_bar,[3,2,1]),n,n,1,n).*((1:n)'==reshape((1:n)',1,1,n));
%
%
%    H=-(aux1+aux2+aux3+aux4);
%    H=permute(H,[2 1 4 3]);
%    hess=reshape(H,n^2,n^2);



    %this version does not require permutation at the last stage

    aux1=(1-rho)*reshape(v,1,n).*reshape(v,1,1,1,n).*reshape(D',n,1,n);
    aux2=reshape(inv(inv_L),n,1,1,n).*reshape(v,1,n).*reshape(term0',1,1,n,n);
%    aux3=permute(aux2,[3 4 1 2]); %permutation is slower
    aux3=reshape(inv(inv_L)',1,n,n).*reshape(v,1,1,1,n).*term0';
    aux4=2*reshape(v,1,n).*permute(A_bar,[2,3,1]).*((1:n)==reshape((1:n)',1,1,1,n));
    H=-(aux1+aux2+aux3+aux4);
    hess=reshape(H,n^2,n^2);

end
```

11.7.15 load_data.m

```
Input Variables
```

```
alpha_data: Data-implied shares from von Lehm and Winberry (2021) (n x n x T array).
tfp_data: Logarithm of TFP data from von Lehm and Winberry (2021) (n x T matrix).
cons: Consumption data from von Lehm and Winberry (2021) (vector or matrix depending on the structure in
    Cons47bea).
price_va: Price value-added data from von Lehm and Winberry (2021) (vector or matrix depending on the
    structure in PriceVA47bea).

 Variables
No explicit output variables are specified here, as the code snippet mainly involves data loading.

Auxiliary Code (External Function Calls)
No external functions are called in this snippet.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%SHARES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Load data from von Lehm and Winberry (2021)
load '../../data_work/Processed Data/ALPHA4720'
alpha_data = ALPHA4720(:,:,2:end);
alpha_data(alpha_data<0)=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%TOTAL FACTOR PRODUCTIVITY%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Load TFP data from von Lehm and Winberry (2021)
dataPath = fullfile(pwd, '../../data_work/Processed Data/TFPMatlab/TFP_GO_nsm_nn.xlsx');
% Verify if the file exists at the constructed path
if exist(dataPath, 'file') ~= 2
   error('The file does not exist at the specified path.');
end
% Read the data using readmatrix
TFP = readmatrix(dataPath, 'Sheet', 1, 'Range', 'B2:AL74');
% Transpose and process the data as in the original script
TFP = TFP';
tfp_data = log(TFP);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%CONSUMPTION%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Load consumption data from von Lehm and Winberry (2021)
load '../../data_work/Processed Data/IOmat4720dat_37sec'
cons=Cons47bea;
price_va=PriceVA47bea;
```

11.7.16 obj_calib_kappa_est.m

```
Input Variables
x: Vector of optimization parameters including elements of kappa_i, kappa_j, and rho.
alpha_bar: Baseline input shares (n x n matrix).
alpha_data: Actual input shares from data (n x n x T array).
tfp_data: Logarithm of TFP data (n x T matrix).
LS_min: Minimum labor supply (scalar).
beta: Vector of discount factors (n x 1 or n x T).
endo_lambda_flag: Flag indicating whether to use endogenous lambda (scalar).
known_shocks_flag: Flag indicating whether to use known shocks (scalar).
flag: Structure with various configuration flags (struct).
cons_gr_data: Consumption growth data (vector).

Output Variables
distance: Scalar measure of the distance between the model and data.

Auxiliary Code (External Function Calls)
estimate_tfp.m: Function to estimate TFP parameters.
compute_eq_time_series.m: Function to compute equilibrium time series.
```

```
function [distance] = obj_calib_kappa_est(x,alpha_bar,alpha_data,tfp_data,LS_min,beta,endo_lambda_flag,
    known_shocks_flag,flag,cons_gr_data)


if isrow(x)
   x =x';
end


```

```
[n,~] = size(tfp_data);
a0=mean(tfp_data,2);

%normalize first element of kappa_i to 15
kappa_i = [15; x(1:n-1)];
kappa_j = x(n:2*n);
kappa = kappa_i*kappa_j';
rho = x(end);

[mu_drift,sigma_t,~,~,epsilon] = estimate_tfp(tfp_data,alpha_bar,alpha_data,kappa,a0,endo_lambda_flag);
[alpha_star,~,C] = compute_eq_time_series(epsilon,mu_drift,sigma_t,kappa,alpha_bar,rho,LS_min,beta,a0,
    known_shocks_flag,flag);
cons_gr_model=C(2:end)-C(1:end-1);

distance = mean((alpha_star-alpha_data).^2,'all')/(mean(abs(alpha_data),'all'))^2 + mean((cons_gr_model-
    mean(cons_gr_model)-cons_gr_data+mean(cons_gr_data)).^2,'all')/(mean(abs(cons_gr_data-mean(
    cons_gr_data)),'all'))^2;



end
```

11.7.17 solve_firm_problem.m

```
Input Variables
mu: Drift term for the firm's problem (n x 1 vector).
sigma: Covariance matrix (n x n matrix).
n: Number of firms (scalar).
rho: Risk aversion parameter (scalar).
LS_min: Minimum labor supply (scalar).
beta: Discount factors (n x 1 or n x T).
i_firm: Index of the current firm (scalar).
alpha_star: Equilibrium input shares (n x n matrix).
a_star: Equilibrium TFP term (n x 1 vector).
A_bar: Matrix A_bar for the firm's problem (n x n matrix).
B_bar: Vector B_bar for the firm's problem (n x 1 vector).
options: Options for the quadprog function (structure).

```

```
Output Variables
alpha_chosen: Chosen input shares for the firm (n x 1 vector).
exitflag: Exit flag from the quadprog function (scalar).

Auxiliary Code (External Function Calls)
quadprog: MATLAB function for quadratic programming.
```

```
function [alpha_chosen,exitflag] = solve_firm_problem(mu,sigma,n,rho,LS_min,beta,i_firm,alpha_star,a_star,A_bar,B_bar,options)
warning('off','optim:quadprog:HessianNotSym')

invL = eye(n,n)-alpha_star;
one_i = zeros(n,1);
one_i(i_firm) = 1;

f = -(B_bar+invL\(mu+a_star-sigma*(one_i-(invL')\(one_i+(1-rho)*beta)))); % Using the dash operator for speed and precision

H = 2*(1/2*(invL\sigma)/(invL')- A_bar); % Using the dash operator for speed and precision


A_const = ones(1,n);
b_const = (1-LS_min);

[alpha_chosen,~,exitflag] = quadprog(H,f,A_const,b_const,[],[],zeros(n,1),ones(n,1)*(1-LS_min),[],options);


if exitflag ~= 1
    options_quad = optimoptions('quadprog','Display','off','Algorithm','active-set');
    alpha_chosen = quadprog(H,f,A_const,b_const,[],[],zeros(n,1),ones(n,1)*(1-LS_min),alpha_star(i_firm,:),options_quad);
end

end
```