ECON 6140

Problem Set 6

Gabe Sekeres

April 3, 2025

1. The household solves the Lagrangian

$$\mathcal{L} = \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \left[U(C_t, N_t) + \lambda_t \left(B_{t-1} + W_t N_t + D_t - P_t C_t - Q_t B_t \right) \right]$$

which admits the first order conditions

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial C_t} &: \beta^t (U_{c,t} - \lambda_t P_t) = 0 \\ \frac{\partial \mathcal{L}}{\partial N_t} &: \beta^t (U_{n,t} - \lambda_t W_t) = 0 \\ \frac{\partial \mathcal{L}}{\partial B_t} &: \beta^{t+1} \mathbb{E}_t \lambda_{t+1} - \beta^t \lambda_t Q_t = 0 \end{aligned}$$

Since we have that $U(C_t, N_t) = \frac{C_t^{1-\sigma} - 1}{1-\sigma} - \frac{N_t^{1+\varphi}}{1+\varphi}$, the household's optimality conditions become

$$\begin{aligned} \frac{N_t^{\varphi}}{C_t^{-\sigma}} &= \frac{W_t}{P_t} \\ C_t^{-\sigma} &= \beta Q_t^{-1} \mathbb{E}_t \left[C_{t+1}^{-\sigma} \frac{P_t}{P_{t+1}} \right] \end{aligned}$$

The firm's optimality condition with the standard production function is, as always,

$$\frac{W_t}{P_t} = (1-\alpha) A_t N_t^{-\alpha}$$

We will log-linearize these conditions, and using the typical lowercase variables, get the optimality conditions:

$$\begin{aligned} \varphi n_t + \sigma c_t &= w_t - p_t \\ c_t &= \mathbb{E}_t[c_{t+1}] - \frac{1}{\sigma} \left(i_t - \rho - \mathbb{E}_t[\pi_{t+1}] \right) \\ w_t - p_t &= \ln(1-\alpha) + a_t - \alpha n_t \end{aligned}$$

Along with the market clearing condition ($y_t = c_t$), this allows us to solve for explicit forms for the the endogenous variables (omitting the arithmetic because it's in the section notes):

$$\begin{aligned} n_t &= \frac{(1-\sigma) a_t + \ln(1-\alpha)}{\varphi + \alpha + \sigma(1-\alpha)} = \psi_{na} a_t + \psi_n \\ y_t &= a_t + (1-\alpha) n_t = \psi_{ya} a_t + \psi_y \\ \omega_t &= \ln(1-\alpha) + a_t - \alpha n_t = \psi_{\omega a} a_t + \psi_\omega \end{aligned}$$

where the ψ variables are as we defined in class; see Table 1

ψ	Expression	(1) Value	(4) Value	(5) Value	(6) Value
ψ_{na}	$\frac{1-\sigma}{\varphi+\alpha+\sigma(1-\alpha)}$	-0.2128	0.137	-0.0855	-0.2222
ψ_{n}	$\frac{\ln(1-\alpha)}{\varphi+\alpha+\sigma(1-\alpha)}$	-0.0759	-0.0977	-0.0305	-0.1540
ψ_{ya}	$\frac{1+\varphi}{\varphi+\alpha+\sigma(1-\alpha)}$	0.8511	1.0959	0.9402	0.8889
ψ_{y}	$\frac{(1-\alpha)\ln(1-\alpha)}{\varphi+\alpha+\sigma(1-\alpha)}$	-0.0531	-0.0684	-0.0213	-0.0770
$\psi_{\omega a}$	$\frac{\varphi+\sigma}{\varphi+\alpha+\sigma(1-\alpha)}$	1.0638	0.9589	1.0256	1.1111
ψ_{ω}	$\frac{(\varphi+\sigma(1-\alpha))\ln(1-\alpha)}{\varphi+\alpha+\sigma(1-\alpha)}$	-0.3339	-0.3274	-0.3475	-0.6161

Table 1: Model Parameterizations (values approximate)

2. ***n.b.*** All code is below in the code section

I simulated 100 periods of a_t, using seed 1234 in Julia for replicability. The plotted time series values are Figure 1

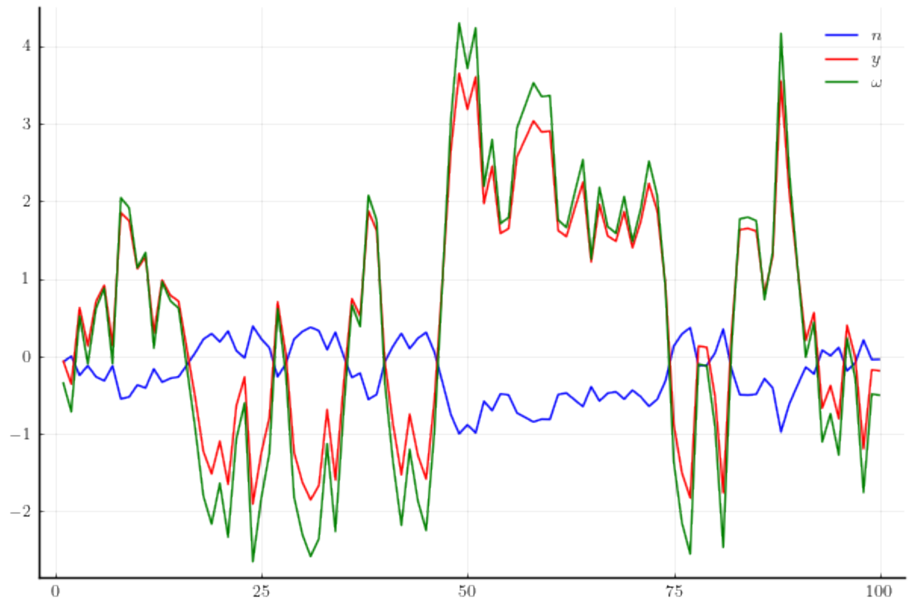

Figure 1: Simulated Economy Under Initial Parameters

The sample variances are

$$(\mathrm{Var}(n), \mathrm{Var}(y), \mathrm{Var}(\omega)) = (0.1322, 2.1154, 3.3054)$$

and the sample cross-variable correlations are

$$\begin{bmatrix} \mathrm{corr}(n,n) & \mathrm{corr}(n,y) & \mathrm{corr}(n,\omega) \\ \mathrm{corr}(y,n) & \mathrm{corr}(y,y) & \mathrm{corr}(y,\omega) \\ \mathrm{corr}(\omega,n) & \mathrm{corr}(\omega,y) & \mathrm{corr}(n,\omega) \end{bmatrix} = \begin{bmatrix} 1.0 & -1.0 & -1.0 \\ -1.0 & 1.0 & 1.0 \\ -1.0 & 1.0 & 1.0 \end{bmatrix}$$

These sample moments are exactly correlated, where even under restrictive assumptions we would not expect the population moments to all be precisely 1. This happens because we assume linearity, and so all of the time series are linear transformations of each other, so they are perfectly correlated.

3. I computed the impulse response functions over 20 periods, and plotted them in Figure 2

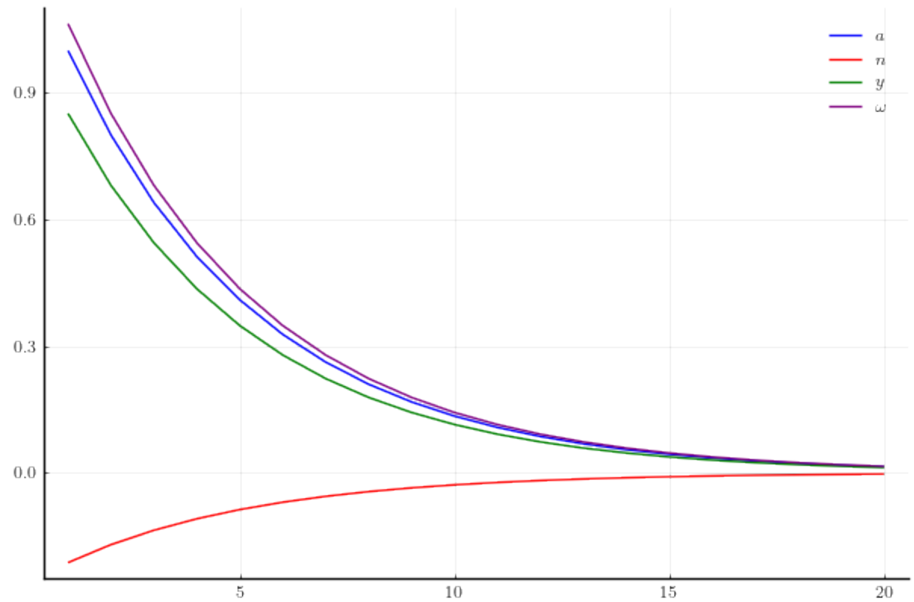

Figure 2: Impulse Response Functions Under Initial Parameters

4. I set $\sigma = 0.5$, and ran the same code as in questions (2) and (3). The simulated economy is plotted in Figure 3

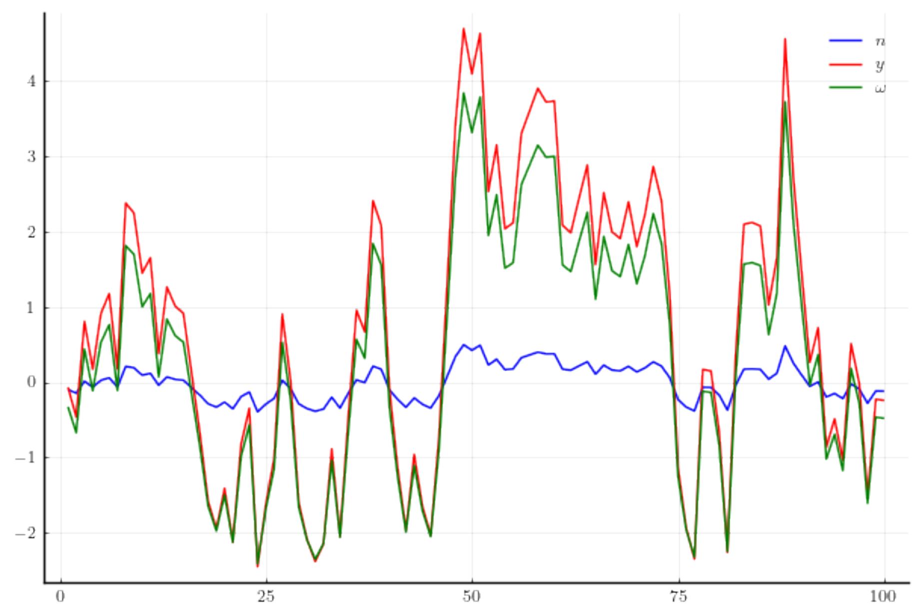

Figure 3: Simulated Economy When $\sigma = 0.5$

The sample variances are

$$(\text{Var}(n), \text{Var}(y), \text{Var}(\omega)) = (0.0548, 3.5076, 2.6855)$$

and the sample cross-variable correlations are

$$\begin{bmatrix} \text{corr}(n,n) & \text{corr}(n,y) & \text{corr}(n,\omega) \\ \text{corr}(y,n) & \text{corr}(y,y) & \text{corr}(y,\omega) \\ \text{corr}(\omega,n) & \text{corr}(\omega,y) & \text{corr}(n,\omega) \end{bmatrix} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Note that now everything is exactly correlated positively! In other words, as the coefficient of relative risk aversion decreases, the household will actually increase their work when their productivity increases, rather than otherwise. The impulse response functions are plotted in Figure 4

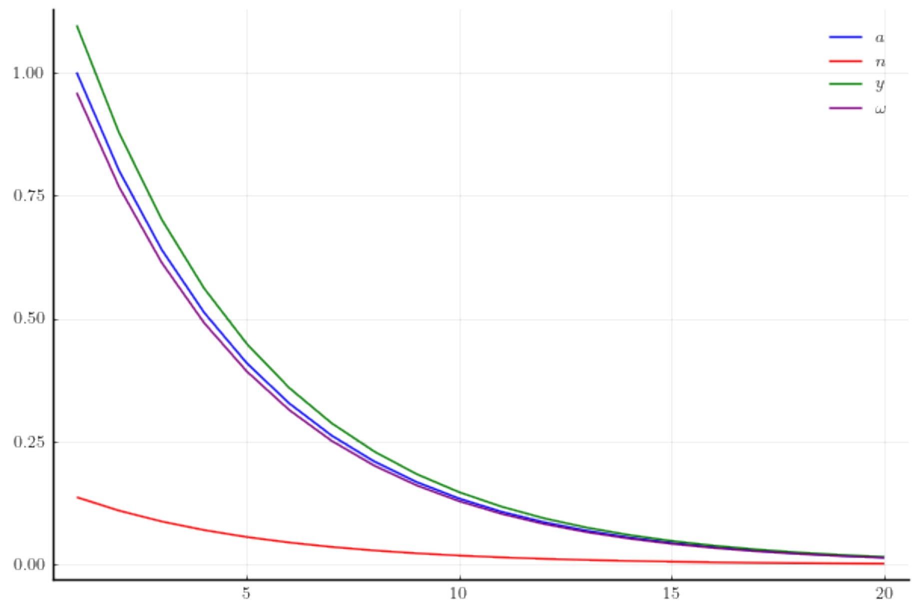

Figure 4: Impulse Response Functions When $\sigma = 0.5$

Note that the variables now move in the same direction.

5. I set $\varphi = 10$, and ran the same code as in questions (2) and (3). The simulated economy is plotted in Figure 5

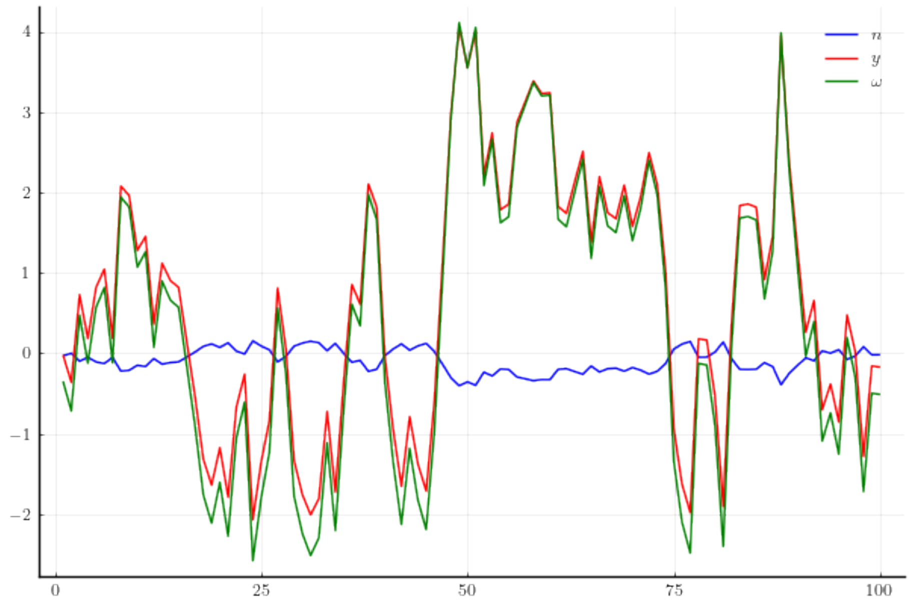

Figure 5: Simulated Economy When $\varphi = 10$

The sample variances are

$$(\mathrm{Var}(n), \mathrm{Var}(y), \mathrm{Var}(\omega)) = (0.0213, 2.5816, 3.0723)$$

and the sample cross-variable correlations are

$$\begin{bmatrix} \mathrm{corr}(n,n) & \mathrm{corr}(n,y) & \mathrm{corr}(n,\omega) \\ \mathrm{corr}(y,n) & \mathrm{corr}(y,y) & \mathrm{corr}(y,\omega) \\ \mathrm{corr}(\omega,n) & \mathrm{corr}(\omega,y) & \mathrm{corr}(n,\omega) \end{bmatrix} = \begin{bmatrix} 1.0 & -1.0 & -1.0 \\ -1.0 & 1.0 & 1.0 \\ -1.0 & 1.0 & 1.0 \end{bmatrix}$$

The impulse response functions are plotted in Figure 6

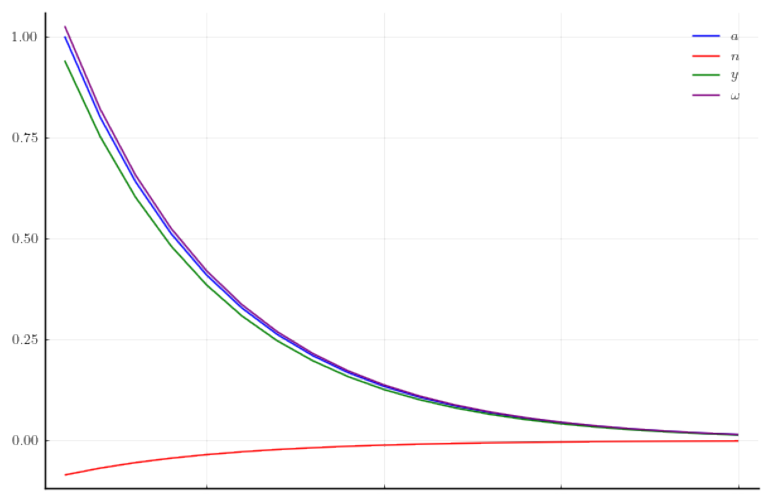

Figure 6: Impulse Response Functions When $\varphi = 10$

Note that now a, y, and ω are extremely closely correlated, even more than in part (2). The interpretation here is that as leisure becomes more valuable, the household works even less, getting closer to the minimum.

6. I set $\alpha = 0.5$, and ran the same code as in questions (2) and (3). The simulated economy is plotted in Figure 7

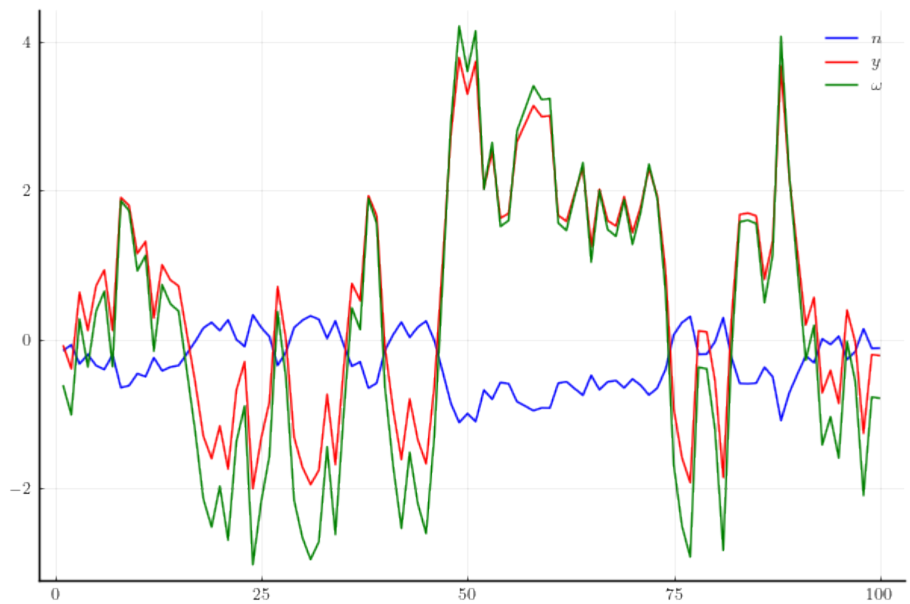

Figure 7: Simulated Economy When $\alpha = 0.5$

The sample variances are

$$(\mathrm{Var}(n), \mathrm{Var}(y), \mathrm{Var}(\omega)) = (0.1442, 2.3077, 3.6057)$$

and the sample cross-variable correlations are

$$\begin{bmatrix} \mathrm{corr}(n,n) & \mathrm{corr}(n,y) & \mathrm{corr}(n,\omega) \\ \mathrm{corr}(y,n) & \mathrm{corr}(y,y) & \mathrm{corr}(y,\omega) \\ \mathrm{corr}(\omega,n) & \mathrm{corr}(\omega,y) & \mathrm{corr}(n,\omega) \end{bmatrix} = \begin{bmatrix} 1.0 & -1.0 & -1.0 \\ -1.0 & 1.0 & 1.0 \\ -1.0 & 1.0 & 1.0 \end{bmatrix}$$

The impulse response functions are plotted in Figure 8

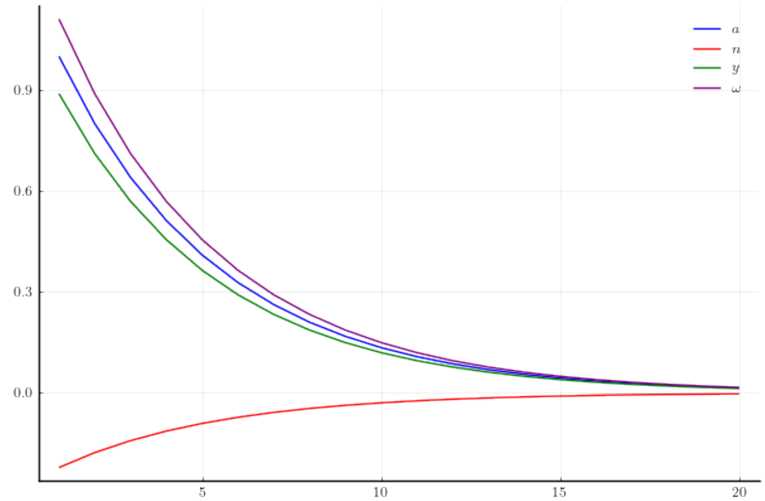

Figure 8: Impulse Response Functions When $\alpha = 0.5$

This is the closest to the initial case, and the interpretation here is that as α increases, workers will care more about current period consumption and less about leisure, as can be seen from their first order conditions.

7. In the main model, the standard deviation of the output gap is std $y_t = 1.4545$, as opposed to the typical standard deviation of 2. If we wanted the standard deviation in this model to match the data, we would change ψ_{ya}, so that a change in a_t would differently change y_t. Specifically, I found that the most straightforward change was to move σ to $\frac{1}{3}$, which led to a standard deviation of 1.936.

Code

I defined the relevant functions in a file called `functions.jl`, which is:

```
mutable struct Parameters
    sigma::Float64
    phi::Float64
    alpha::Float64
    rhoalpha::Float64
    sigmaalpha::Float64
    beta::Float64
    psina::Float64
    psin::Float64
    psiya::Float64
    psiy::Float64
    psiomegaa::Float64
    psiomega::Float64

    function Parameters(;sigma=2.0, phi=3.0, alpha=0.3, rhoalpha=0.8,
       sigmaalpha=1.0, beta=0.99)
        denom = phi + alpha + sigma * (1 - alpha)
        psina = (1 - sigma) / denom
        psin = (log(1 - alpha)) / denom
        psiya = (1 + phi) / denom
        psiy = (1 - alpha) * psin
        psiomegaa = (phi + sigma) / denom
        psiomega = ((phi + sigma * (1 - alpha)) * log(1 - alpha)) / denom
        return new(sigma, phi, alpha, rhoalpha, sigmaalpha, beta, psina, psin,
            psiya, psiy, psiomegaa, psiomega)
    end
end
"""
    simulate(p::Parameters, T::Int; seed=...) -> Dict

Simulate T periods of the model. Returns a Dict with vectors of
{a, n, y, omega}. The user can specify a seed for reproducibility.
"""
function simulate(p::Parameters, T::Int; seed::Int=0)
    if seed != 0
        Random.seed!(seed)
    end
    a = zeros(T)
    n = zeros(T)
    y = zeros(T)
    omega = zeros(T)

    # For the shock process a_{t+1} = rho * a_t + e_{t+1}
    # with e_{t+1} ~ Normal(0, sigmaalpha^2).
    for t in 2:T
        shock = randn() * sqrt(p.sigmaalpha)
        a[t] = p.rhoalpha * a[t-1] + shock
    end
```

```julia
    # Now compute n, y, omega from the closed-form solutions:
    for t in 1:T
        n[t]      = p.psina    * a[t] + p.psin
        y[t]      = p.psiya    * a[t] + p.psiy
        omega[t]  = p.psiomegaa* a[t] + p.psiomega
    end

    return Dict(:a => a, :n => n, :y => y, :omega => omega)
end
"""
    compute_sample_moments(data::Dict) -> Dict

Given a Dict of time series (key => vector), compute:
- Variances
- Cross-correlations

Returns another Dict with those statistics.
"""
function compute_sample_moments(data::Dict)
    a = data[:a]
    n = data[:n]
    y = data[:y]
    omega = data[:omega]

    va = var(a)
    vn = var(n)
    vy = var(y)
    vomega = var(omega)

    M = hcat(n, y, omega)
    C = cor(Matrix(M))

    return Dict(
        :var => (vn, vy, vomega),
        :corr => C
    )
end

"""
    compute_irf(p::Parameters, horizon::Int) -> Dict

Computes impulse responses of n_t, y_t, omega_t to a one-time shock in
   epsilon_a.
The approach here:
1. Start with a(0) = 0, then at t=1 let epsilon_a = 1 (one-unit shock),
   for t>1 all shocks = 0.
2. Record a(t), then compute n(t), y(t), omega(t).
3. Return the deviations from the no-shock baseline.
"""
function compute_irf(p::Parameters, horizon::Int)
```

```julia
    a_irf = zeros(horizon)
    n_irf = zeros(horizon)
    y_irf = zeros(horizon)
    omega_irf = zeros(horizon)

    for t in 1:horizon
        a_current = p.rhoalpha^(t-1) * 1.0

        # Full levels with shock:
        n_shock = p.psina*a_current + p.psin
        y_shock = p.psiya*a_current + p.psiy
        omega_shock = p.psiomegaa*a_current + p.psiomega

        # Baseline levels (no shock) are:
        n_irf[t] = n_shock - p.psin
        y_irf[t] = y_shock - p.psiy
        omega_irf[t] = omega_shock - p.psiomega
        a_irf[t] = a_current
    end

    return Dict(:a => a_irf, :n => n_irf, :y => y_irf, :omega => omega_irf)
end
```

I ran the rest of the code from a file called `main.jl`, which is:

```
using Random, Statistics, Plots
include("functions.jl")

# Make the plots look pretty
pyplot()
PyPlot.rc("text", usetex=true)
PyPlot.rc("font", family="serif")
PyPlot.matplotlib.rcParams["mathtext.fontset"] = "cm"

# Preliminaries
println("Check different parameters:")
parameters1 = Parameters()
parameters4 = Parameters(sigma = 0.5)
parameters5 = Parameters(phi = 10)
parameters6 = Parameters(alpha = 0.5)

# Simulation horizon
T = 100

println("Initial: sigma =$(parameters1.sigma) & phi = $(parameters1.phi) &
    alpha = $(parameters1.alpha) & rhoalpha = $(parameters1.rhoalpha) &
    sigmaalpha = $(parameters1.sigmaalpha) & beta = $(parameters1.beta)")
println("Part 4: sigma = $(parameters4.sigma) & phi = $(parameters4.phi) &
    alpha = $(parameters4.alpha) & rhoalpha = $(parameters4.rhoalpha) &
    sigmaalpha = $(parameters4.sigmaalpha) & beta = $(parameters4.beta)")
println("Part 5: sigma = $(parameters5.sigma) & phi = $(parameters5.phi) &
    alpha = $(parameters5.alpha) & rhoalpha = $(parameters5.rhoalpha) &
    sigmaalpha = $(parameters5.sigmaalpha) & beta = $(parameters5.beta)")
println("Part 6: sigma = $(parameters6.sigma) & phi = $(parameters6.phi) &
    alpha = $(parameters6.alpha) & rhoalpha = $(parameters6.rhoalpha) &
    sigmaalpha = $(parameters6.sigmaalpha) & beta = $(parameters6.beta)")
println("psina: $(round(parameters1.psina, digits=4)) & $(round(parameters4.
    psina, digits=4)) & $(round(parameters5.psina, digits=4)) & $(round(
    parameters6.psina, digits=4))")
println("psin: $(round(parameters1.psin, digits=4)) & $(round(parameters4.psin
    , digits=4)) & $(round(parameters5.psin, digits=4)) & $(round(parameters6.
    psin, digits=4))")
println("psiya: $(round(parameters1.psiya, digits=4)) & $(round(parameters4.
    psiya, digits=4)) & $(round(parameters5.psiya, digits=4)) & $(round(
    parameters6.psiya, digits=4))")
println("psiy: $(round(parameters1.psiy, digits=4)) & $(round(parameters4.psiy
    , digits=4)) & $(round(parameters5.psiy, digits=4)) & $(round(parameters6.
    psiy, digits=4))")
println("psiomegaa: $(round(parameters1.psiomegaa, digits=4)) & $(round(
    parameters4.psiomegaa, digits=4)) & $(round(parameters5.psiomegaa, digits
    =4)) & $(round(parameters6.psiomegaa, digits=4))")
println("psiomega: $(round(parameters1.psiomega, digits=4)) & $(round(
    parameters4.psiomega, digits=4)) & $(round(parameters5.psiomega, digits=4)
    ) & $(round(parameters6.psiomega, digits=4))")
```

```
# Question 2
simdata1 = simulate(parameters1, T, seed=1234)
varcorr1 = compute_sample_moments(simdata1)
println("variances (n, y, omega): $(round.(varcorr1[:var], digits=4))")
println("correlations (n, y, omega) x (n, y, omega): $(round.(varcorr1[:corr],
    digits=4))")
n_data1 = simdata1[:n]
y_data1 = simdata1[:y]
omega_data1 = simdata1[:omega]

p11 = plot(background=:transparent)
plot!(p11, n_data1, label="\$n\$", color=:blue)
plot!(p11, y_data1, label="\$y\$", color=:red)
plot!(p11, omega_data1, label="\$\\omega\$", color=:green)

savefig(p11, "macro_hw6_code/q2_simdata.png")

irf1 = compute_irf(parameters1, 20)

p12 = plot(background=:transparent)
plot!(p12, irf1[:a], label="\$a\$", color=:blue)
plot!(p12, irf1[:n], label="\$n\$", color=:red)
plot!(p12, irf1[:y], label="\$y\$", color=:green)
plot!(p12, irf1[:omega], label="\$\\omega\$", color=:purple)

savefig(p12, "macro_hw6_code/q2_irf.png")


# Question 4
simdata4 = simulate(parameters4, T, seed=1234)
varcorr4 = compute_sample_moments(simdata4)
println("variances (n, y, omega): $(round.(varcorr4[:var], digits=4))")
println("correlations (n, y, omega) x (n, y, omega): $(round.(varcorr4[:corr],
    digits=4))")
n_data4 = simdata4[:n]
y_data4 = simdata4[:y]
omega_data4 = simdata4[:omega]

p41 = plot(background=:transparent)
plot!(p41, n_data4, label="\$n\$", color=:blue)
plot!(p41, y_data4, label="\$y\$", color=:red)
plot!(p41, omega_data4, label="\$\\omega\$", color=:green)

savefig(p41, "macro_hw6_code/q4_simdata.png")

irf4 = compute_irf(parameters4, 20)

p42 = plot(background=:transparent)
plot!(p42, irf4[:a], label="\$a\$", color=:blue)
plot!(p42, irf4[:n], label="\$n\$", color=:red)
```

```
plot!(p42, irf4[:y], label="\$y\$", color=:green)
plot!(p42, irf4[:omega], label="\$\\omega\$", color=:purple)

savefig(p42, "macro_hw6_code/q4_irf.png")

# Question 5
simdata5 = simulate(parameters5, T, seed=1234)
varcorr5 = compute_sample_moments(simdata5)
println("variances (n, y, omega): $(round.(varcorr5[:var], digits=4))")
println("correlations (n, y, omega) x (n, y, omega): $(round.(varcorr5[:corr],
    digits=4))")
n_data5 = simdata5[:n]
y_data5 = simdata5[:y]
omega_data5 = simdata5[:omega]

p51 = plot(background=:transparent)
plot!(p51, n_data5, label="\$n\$", color=:blue)
plot!(p51, y_data5, label="\$y\$", color=:red)
plot!(p51, omega_data5, label="\$\\omega\$", color=:green)

savefig(p51, "macro_hw6_code/q5_simdata.png")

irf5 = compute_irf(parameters5, 20)

p52 = plot(background=:transparent)
plot!(p52, irf5[:a], label="\$a\$", color=:blue)
plot!(p52, irf5[:n], label="\$n\$", color=:red)
plot!(p52, irf5[:y], label="\$y\$", color=:green)
plot!(p52, irf5[:omega], label="\$\\omega\$", color=:purple)

savefig(p52, "macro_hw6_code/q5_irf.png")

# Question 6
simdata6 = simulate(parameters6, T, seed=1234)
varcorr6 = compute_sample_moments(simdata6)
println("variances (n, y, omega): $(round.(varcorr6[:var], digits=4))")
println("correlations (n, y, omega) x (n, y, omega): $(round.(varcorr6[:corr],
    digits=4))")
n_data6 = simdata6[:n]
y_data6 = simdata6[:y]
omega_data6 = simdata6[:omega]

p61 = plot(background=:transparent)
plot!(p61, n_data6, label="\$n\$", color=:blue)
plot!(p61, y_data6, label="\$y\$", color=:red)
plot!(p61, omega_data6, label="\$\\omega\$", color=:green)

savefig(p61, "macro_hw6_code/q6_simdata.png")

irf6 = compute_irf(parameters6, 20)
```

```
p62 = plot(background=:transparent)
plot!(p62, irf6[:a], label="\$a\$", color=:blue)
plot!(p62, irf6[:n], label="\$n\$", color=:red)
plot!(p62, irf6[:y], label="\$y\$", color=:green)
plot!(p62, irf6[:omega], label="\$\\omega\$", color=:purple)

savefig(p62, "macro_hw6_code/q6_irf.png")


# Question 7
println("STD y: $(round(std(y_data1), digits=4))")

parameters_y = Parameters(sigma=0.33)
simdata_y = simulate(parameters_y, T, seed=1234)
println("STD y: $(round(std(simdata_y[:y]), digits=4))")
```