Dynamic Rational Inattention Problems (DRIPs)[*]

Hassan Afrouzi[†]
Columbia University

Choongryul Yang[‡]
Federal Reserve Board

March 28, 2021

Abstract

This document provides a hands-on introduction to `DRIPs.jl`, which is a Julia software package that provides a fast and robust method for solving LQG Dynamic Rational Inattention models using the methods developed by Afrouzi and Yang (2019). The layout and the content of this document closely follows the documentation released with `DRIPs.jl`, available at `https://afrouzi.com/DRIPs.jl/dev/`. A twin Matlab package is available at `https://github.com/choongryulyang/DRIPs.m`.

Contents

[*]The views expressed here are those of the authors and do not necessarily reflect those of the Federal Reserve Board or the Federal Reserve System. First version: Feb 2021.

[†]Department of Economics, Columbia University, 420 W 118th Street, New York, NY 10027, U.S.A. Email: ha2475@columbia.edu .

[‡]Federal Reserve Board of Governors, 20th Street and Constitution Avenue NW, Washington, DC 20551, U.S.A. Email: choongryul.yang@frb.gov.

1 Overview

1.1 Installation

To add the package, execute the following in Julia REPL:

```
using Pkg; Pkg.add("DRIPs");
```

To import and use the package, execute:

```
using DRIPs;
```

1.2 Definition of the Problem

A LQG dynamic rational inattention problem (DRIP) is defined as the following, where at any point in time an agent chooses a vector of actions $\vec{a}_t \in \mathbb{R}^m$ to track a Gaussian stochastic process $\vec{x}_t \in \mathbb{R}^n$:

$$\begin{aligned} \min_{\{\vec{a}_t\}_{t\geq 0}} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t (\vec{a}_t - \vec{x}_t'\mathbf{H})'(\vec{a}_t - \vec{x}_t'\mathbf{H}) + \omega \mathbb{I}(\vec{a}^t;\vec{x}^t|\vec{a}^{t-1})|\vec{a}^{-1}\right] \\ s.t. \quad & \vec{x}_t = \mathbf{A}\vec{x}_{t-1} + \mathbf{Q}\vec{u}_t, \quad \vec{u}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}^{k\times k}) \\ & \vec{a}^{-1} \text{ given.} \end{aligned}$$

Here:

- $\vec{a}_t \in \mathbb{R}^m$ is the vector of the agent's actions at time t (a firms choosing a price, or a househld choosing consumption and labor etc.) We denote the number of actions by m.
- $\vec{x}_t \in \mathbb{R}^n$ is the vector of the shocks that the agent faces at time t that are exogenous to her decision, but could be endogenous to the GE model (marginal cost of a firm, real interest rates etc.) We denote the number of shocks by n.

Such quadratic payoff functions can be derived as a second-order approximation to a general twice differentiable function $v(\vec{a}_t;\vec{x}_t)$ and applying a proper change of variables (for instance, normalizing the Hessian of v with respect to a to identity, or in case of endogenous state variables, writing the innovation to the agent's action in terms of the agent's belief about $\vec{x}_t$).[1]

Parameters A DRIP is characterized by the following parameters:

[1]See, for instance, Mackowiak and Wiederholt (2020) for such a treatment of endogenous state variables.

- $\omega \in \mathbb{R}_+$: cost of 1 bit of information in units of the agent's payoff.
- $\beta \in [0,1]$: rate of discounting information.
- $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{Q} \in \mathbb{R}^{n\times k}$: Determine the state space representation of $\vec{x}_t$.
- $\mathbf{H} \in \mathbb{R}^{n\times m}$: interaction of payoffs with shocks. This comes from a second order approximation to the utility function and is such that under full information $\vec{a}^* = \mathbf{H}'\vec{x}$.

Solution The solution to the dynamic rational inattention problem is a joint stochastic process between the actions and the states: $\{(\vec{a}_t, \vec{x}_t) : t \geq 0\}$. Moreover, in some economic applications, we are also interested in the law of motion for the agent's belief about $\vec{x}_t$ under the optimal information structure $\hat{x}_t = \mathbb{E}_t[\vec{x}_t]$ where the expectation is taken conditional on the agent's time t information.

Theorem 2.2 and Proposition 2.3 in Afrouzi and Yang (2019) characterize this joint distribution as a function of a tuple $(\mathbf{K}_t, \mathbf{Y}_t, \mathbf{\Sigma}_{z,t})$ where

$$\begin{aligned}\vec{a}_t &= \mathbf{H}'\hat{x}_t = \mathbf{H}'\mathbf{A}\hat{x}_{t-1} + \mathbf{Y_t}'(\vec{x}_t - \mathbf{A}\hat{x}_{t-1}) + \vec{z}_t \\ \hat{x}_t &= \mathbf{A}\hat{x}_{t-1} + \mathbf{K}_t\mathbf{Y}_t'(\vec{x}_t - \mathbf{A}\hat{x}_{t-1}) + \mathbf{K}_t\vec{z}_t, \quad \vec{z} \sim \mathcal{N}(0, \Sigma_z)\end{aligned}$$

Here,

- $\mathbf{K}_t \in \mathbb{R}^{n\times m}$ is the Kalman-gain matrix of the agent in under optimal information acquisition at time t.
- $\mathbf{Y}_t \in \mathbb{R}^{m\times m}$ is the loading of optimal signals on the state at time t.
- $\mathbf{\Sigma}_{z,t} \in \mathbb{R}^{m\times m}$ is the variance-covariance matrix of the agent's rational inattention error at time t.

In addition to these, we might also be interested in the agent's prior and posterior subjective uncertainty, along with the continuation value that she assigns to information:

- $\mathbf{\Sigma}_{p,t} = \mathbb{V}ar(\vec{x}_t|\vec{a}^t) \in \mathbb{R}^{n\times n}$.
- $\mathbf{\Sigma}_{-1,t} = \mathbb{V}ar(\vec{x}_t|\vec{a}^{t-1}) \in \mathbb{R}^{n\times n}$,
- $\mathbf{\Omega}_t \in \mathbb{R}^{n\times n}$.

where the matrix $\mathbf{\Omega}_t$ captures the value of information (see Afrouzi and Yang (2019) for details).

1.3 Steady State of DRIPs

Use function `Drip`(ω, β, A, Q, H) to store and solve for the steady state of a DRIP. It takes the primitives (ω, β, A, Q, H) as arguments and returns the solution of the model within a `Drip` type structure that (1) stores all the primitives of the problem and (2) returns the steady state solution information structure of the model in a field called `ss` that is of type `DRIPs.SteadyState`.

1.4 Transition Dynamics of DRIPs

The Euler equation derived in Afrouzi and Yang (2019) for the also allows us to characterize the transition path of the information structure over time for an arbitrary initial prior.

The function `Trip(P::Drip,s0)` takes a `Drip` type structure along with an initial condition `s0` as an input and returns a `Trip` structure that summarizes the transition path of the optimal information structure. The initial condition `s0` can be given either as an initial prior covariance matrix or alternatively as a one time signal about the state that perturbs the steady state prior.

1.5 Impulse Response Functions

Once the model is solved, one can generate the impulse response functions of actions and beliefs using the laws of motion stated above.

We have also included a built-in function that generates these IRFs. The function `irfs(P::Drip)` takes a `Drip` structure as input and returns the irfs of the state, beliefs and actions to all structural shocks within a `Path` structure.

The function also returns IRFs for transition dynamics if an initial signal is specified.

1.6 Simulations of DRIPs

Once the model is solved, one can also generate the simulated paths for fundamental, actions and beliefs..

We have also included a built-in function that generates these IRFs. The function `simulate(P::Drip)` takes a `Drip` structure as input and returns a simulated path of the state, beliefs and actions to all structural shocks within a `Path` structure.

2 Examples and Replications

2.1 Pricing under RI without Endogenous Feedback

This document goes through a couple of examples for solving pricing under rational inattention without endogenous feedback using the `DRIPs` package. See Afrouzi and Yang (2019) for background on the theory.

2.1.1 One Shock Case

There is a measure of firms indexed by $i \in [0,1]$. Firm i chooses its price $p_{i,t}$ at time t to track its ideal price $p^*_{i,t}$. Formally, her flow profit is

$$-(p_{i,t} - p^*_{i,t})^2.$$

We start by assuming that $p^*_{i,t} = q_t$ where

$$\Delta q_t = \rho \Delta q_{t-1} + u_t, \quad u_t \sim \mathcal{N}(0, \sigma_u^2)$$

Here q_t can be interpreted as money growth or the nominal aggregate demand. Therefore, the state-space representation of the problem is

$$\vec{x}_t = \begin{bmatrix} q_t \\ \Delta q_t \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \rho \\ 0 & \rho \end{bmatrix}}_{\mathbf{A}} \vec{x}_{t-1} + \underbrace{\begin{bmatrix} \sigma_u \\ \sigma_u \end{bmatrix}}_{\mathbf{Q}} u_t,$$

$$p^*_{i,t} = \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}'}_{\mathbf{H}} \vec{x}_t$$

The following instructions implement this model in DRIPs.

Include the package:

```
using DRIPs;
```

Assign value to deep parameters and define the structure of the problem:

```
ρ   = 0.6;        #persistence of money growth
σ_u = 1;          #std. deviation of shocks to money growth
```

Primitives of the DRIP:

```
ω   = 100;
β   = 0.96^0.25;
A   = [1 ρ; 0 ρ];
Q   = σ_u*[1; 1];
H   = [1; 0];
```

Get solution:

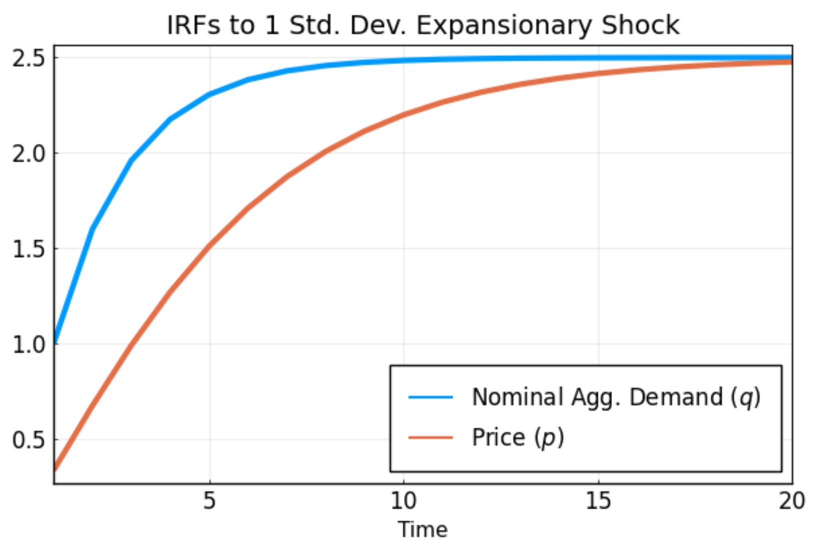
```
ex1  = Drip(ω,β,A,Q,H);
```

Get IRFs:

```
ex1irfs = irfs(ex1, T=20);
```

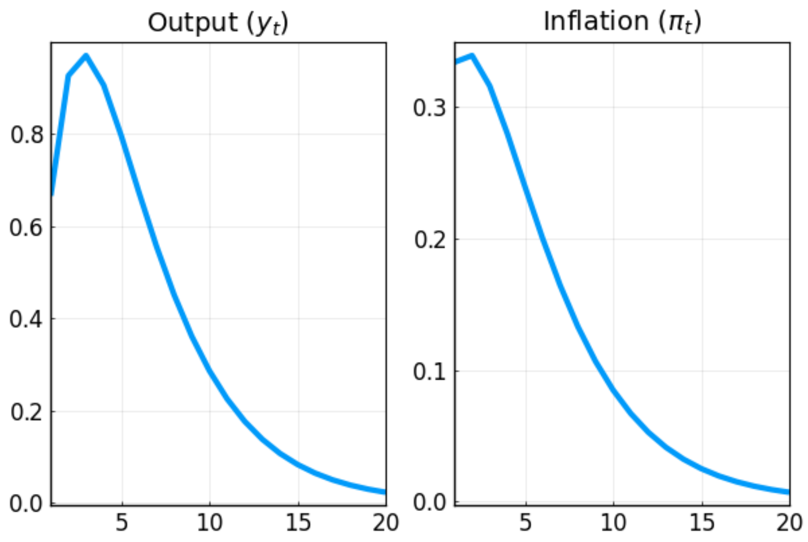
Let's plot how the average price $p = \int_0^1 p_{i,t} di$ responds to a shock to money growth:

```
using Plots, LaTeXStrings; pyplot();
plot(1:ex1irfs.T,[ex1irfs.x[1,1,:],ex1irfs.a[1,1,:]],
    xlabel     = "Time",
    label      = [L"Nominal Agg. Demand ($q$)" L"Price ($p$)"],
    title      = "IRFs to 1 Std. Dev. Expansionary Shock",
    xlim       = (1,ex1irfs.T),
    lw         = 3,
    legend     = :bottomright,
    legendfont = font(12),
    tickfont   = font(12),
    framestyle = :box)}
```



We can also plot the IRFs of inflation $\pi_t \equiv p_t - p_{t-1}$ and output $y_t \equiv q_t - p_t$ to 1 percent expansionary shock to q:

```
p1 = plot(1:ex1irfs.T,
          ex1irfs.x[1,1,:]-ex1irfs.a[1,1,:],
     title  = L"Output ($y_t$)")
p2 = plot(1:ex1irfs.T,
          [ex1irfs.a[1,1,1];ex1irfs.a[1,1,2:end]-ex1irfs.a[1,1,1:end-1]],
     title  = L"Inflation ($\pi_t$)")
plot(p1,p2,
    layout     = (1,2),
    xlim       = (1,ex1irfs.T),
    lw         = 3,
    legend     = false,
    tickfont   = font(12),
    framestyle = :box)
```



2.1.2 Two Shocks Case

Suppose now that $p^*_{i,t} = q_t - z_t$ where

$$\begin{aligned} \Delta q_t &= \rho \Delta q_{t-1} + u_t, \quad u_t \sim \mathcal{N}(0, \sigma_u^2) \\ z_t &\sim \mathcal{N}(0, \sigma_z^2) \end{aligned}$$

Here q_t can be interpreted as money growth and $z_{i,t}$ as an idiosyncratic TFP shock. Therefore,

$$\begin{aligned} \vec{x}_t &= \begin{bmatrix} q_t \\ \Delta q_t \\ z_t \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \rho & 0 \\ 0 & \rho & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \vec{x}_{t-1} + \underbrace{\begin{bmatrix} \sigma_u & 0 \\ \sigma_u & 0 \\ 0 & \sigma_z \end{bmatrix}}_{\mathbf{Q}} \begin{bmatrix} u_t \\ z_t \end{bmatrix}, \\ p^*_{i,t} &= \underbrace{\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}}_{\mathbf{H}}{}' \vec{x}_t \end{aligned}$$

The following instructions implement this model in DRIPs.

Assign values:

```
ρ   = 0.6;        #persistence of money growth
σ_u = 1;          #std. deviation of shocks to money growth
σ_z = 10^0.5;     #std. deviation of idiosyncratic shock
```

Primitives of the DRIP:

```
ω   = 100;
β   = 0.96^0.25;
A   = [1 ρ 0; 0 ρ 0; 0 0 0];
Q   = [σ_u 0; σ_u 0; 0 σ_z];
H   = [1; 0; -1];
```

Get solution:

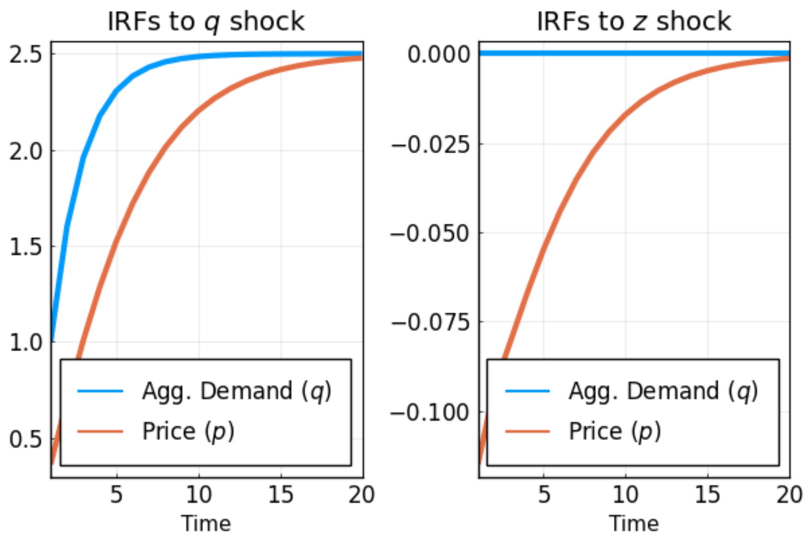
```
ex2  = Drip(ω,β,A,Q,H);
```

Get IRFs:

```
ex2irfs = irfs(ex2, T=20);
```

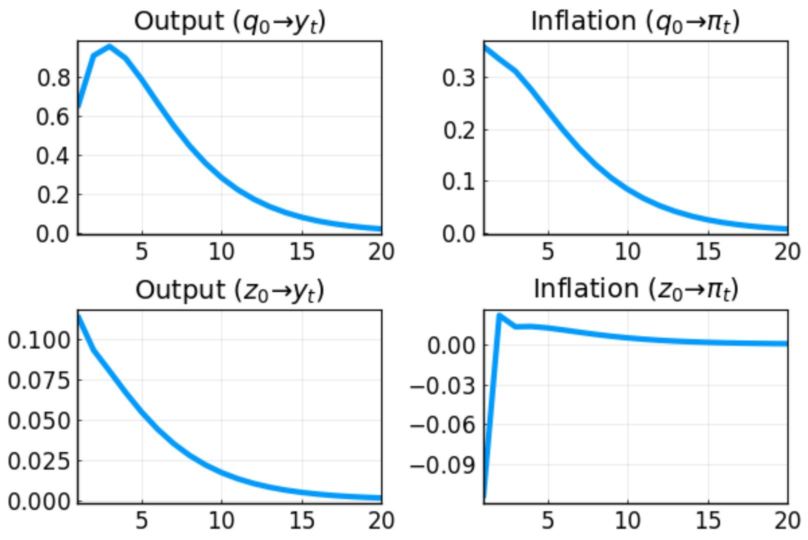
To get the IRFs simply use the law of motion for actions:

```
p1 = plot(1:ex2irfs.T,[ex2irfs.x[1,1,:],ex2irfs.a[1,1,:]],
     title  = L"IRFs to $q$ shock");
p2 = plot(1:ex1irfs.T,[ex2irfs.x[1,2,:],ex2irfs.a[1,2,:]],
     title  = L"IRFs to $z$ shock");
plot(p1,p2, layout = (1,2),
     xlabel     = "Time", xlim = (1,ex2irfs.T),
     label      = [L"Agg. Demand ($q$)" L"Price ($p$)"],
     lw         = 3,
     legend     = :bottomright,
     legendfont = font(12), tickfont = font(12),
     framestyle = :box)
```



More IRFs:

```
p1 = plot(1:ex2irfs.T,ex2irfs.x[1,1,:]-ex2irfs.a[1,1,:],
     title  = L"Output ($q_0\to y_t$)");
p2 = plot(1:ex2irfs.T,
     [ex2irfs.a[1,1,1];ex2irfs.a[1,1,2:end]-ex2irfs.a[1,1,1:end-1]],
     title  = L"Inflation ($q_0\to \pi_t$)")
p3 = plot(1:ex2irfs.T,
     ex2irfs.x[1,2,:]-ex2irfs.a[1,2,:],
     title  = L"Output ($z_0\to y_t$)");
p4 = plot(1:ex2irfs.T,
     [ex2irfs.a[1,2,1];ex2irfs.a[1,2,2:end]-ex2irfs.a[1,2,1:end-1]],
     title  = L"Inflation ($z_0\to \pi_t$)")

plot(p1,p2,p3,p4, layout = (2,2),
     xlim       = (1,ex2irfs.T),
     lw         = 3,
     legend     = false,
     tickfont   = font(12),
     framestyle = :box)
```



2.2 Pricing under RI with Endogenous Feedback

This example solves a pricing problem under rational inattention with endogenous feedback using the `DRIPs` package.

2.2.1 Setup

Problem Suppose now that there is general equilibrium feedback with the degree of strategic complementarity α:

$$p_{i,t}^* = (1-\alpha)q_t + \alpha p_t$$

where

$$\Delta q_t = \rho \Delta q_{t-1} + u_t, \quad u_t \sim \mathcal{N}(0, \sigma_u^2)$$
$$p_t \equiv \int_0^1 p_{i,t} di$$

Note that now the state space representation for $p_{i,t}^*$ is no longer exogenous and is determined in the equilibrium. However, we know that this is a Guassian process and by Wold's theorem we can decompose it to its $MA(\infty)$ representation:

$$p_{i,t}^* = \Phi(L)u_t$$

where $\Phi(.)$ is a lag polynomial and u_t is the shock to nominal demand. Here, we have basically guessed that the process for $p_{i,t}^*$ is determined uniquely by the history of monetary shocks which requires that rational inattention errors of firms are orthogonal (See Afrouzi, 2020). Our objective is to find $\Phi(.)$.

Since we cannot put $MA(\infty)$ processes in the computer, we approximate them with truncation. In particular, we know for stationary processes, we can arbitrarily get close to the true process by truncating $MA(\infty)$ processes to $MA(T)$ processes. Our problem here is that $p_{i,t}^*$ has a unit root and is not stationary. We can bypass this issue by re-writing the state space in the following way:

$$p_{i,t}^* = \phi(L)\tilde{u}_t, \quad \tilde{u}_t = (1-L)^{-1}u_t = \sum_{j=0}^{\infty} u_{t-j}$$

where $\tilde{u}_{t-j}$ is the unit root of the process and basically we have differenced out the unit root from the lag polynomial, and $\phi(L) = (1-L)\Phi(L)$. Notice that since the original process was difference stationary, differencing out the unit root means that $\phi(L)$ is now in ℓ_2, and the process can now be approximated arbitrarily precisely with truncation.

Matrix Notation For a length of truncation L, let $\vec{x}t \equiv (\tilde{u}t, \tilde{u}t-1, \dots, \tilde{u}t-(L+1)) \in \mathbb{R}^L$. Then, note that $p_{i,t}^* \approx \mathbf{H}'\vec{x}_t$ where $\mathbf{H} \in \mathbb{R}^L$ is the truncated matrix analog of the lag polynominal, and is endogenous to the problem. Our objective is to find the general equilibrium $\mathbf{H}$ along with the optimal information structure that it implies.

Moreover, note that

$$q_t = \mathbf{H}_q' \vec{x}_t, \quad \mathbf{H}_q' = (1, \rho, \rho^2, \dots, \rho^{L-1})$$

We will solve for ϕ by iterating over the problem. In particular, in iteration $n \geq 1$, given the guess $\mathbf{H}_{(n-1)}$, we have the following state space representation for the firm's problem:

$$\vec{x}_t = \underbrace{\begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}}_{\mathbf{A}} \vec{x}_{t-1} + \underbrace{\begin{bmatrix} \sigma_u \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{\mathbf{Q}} u_t,$$

$$p_{i,t}^* = \mathbf{H}_{(n-1)}' \vec{x}_t$$

Then we can solve the rational inattention problem of all firms and get the new guess for p_t^*:

$$\begin{aligned} p_t^* &= (1-\alpha) q_t + \alpha p_t \\ &= (1-\alpha) \sum_{j=0}^{\infty} \alpha^j q_t^{(j)}, \\ q_t^{(j)} &\equiv \begin{cases} q_t & j = 0 \\ \int_0^1 \mathbb{E}_{i,t}[q_t^{(j-1)}] di & j \geq 1 \end{cases} \end{aligned}$$

where $q_t^{(j)}$ is the j'th order belief of firms, on average, of q_t. Now, we need to write these higher order beliefs in terms of the state vector. Suppose, for a given j, there exists $\mathbf{X}_j \in \mathbb{R}^{L \times L}$ such that

$$q_t^{(j)} = \mathbf{H}_q' \mathbf{X}_j \vec{x}_t$$

This clearly holds for $j = 0$ with $\mathbf{X}_0 = \mathbf{I}$.

Now, note that

$$
\begin{aligned}
q_t^{(j+1)} &= \int_0^1 \mathbb{E}_{i,t}[q_t^{(j)}]di \\
&= \mathbf{H}_q'\mathbf{X}_j \int_0^1 \mathbb{E}_{i,t}[\vec{x}_t]di \\
&= \mathbf{H}_q'\mathbf{X}_j \sum_{j=0}^{\infty}[(\mathbf{I}-\mathbf{K}_{(n)}\mathbf{Y}_{(n)}')\mathbf{A}]^j\mathbf{K}_{(n)}\mathbf{Y}_{(n)}'\vec{x}_{t-j} \\
&\approx \mathbf{H}_q'\mathbf{X}_j \underbrace{\left[\sum_{j=0}^{\infty}[(\mathbf{I}-\mathbf{K}_{(n)}\mathbf{Y}_{(n)}')\mathbf{A}]^j\mathbf{K}_{(n)}\mathbf{Y}_{(n)}'\mathbf{M}'^j\right]}_{\equiv\mathbf{X}_{(n)}}\vec{x}_t = \mathbf{H}_q'\mathbf{X}_j\mathbf{X}_{(n)}\vec{x}_t
\end{aligned}
$$

where the (n) subscripts refer to the solution of the RI problem in the (n)'th iteration. Note that this implies

$$
\mathbf{X}_j = \mathbf{X}_{(n)}^j, \forall j \geq 0 \Rightarrow q_t^{(j)} = \mathbf{X}_{(n)}^j\vec{x}_t
$$

This gives us an updated guess for $\mathbf{H}$:

$$
p_t^* = (1-\alpha)\mathbf{H}_q' \underbrace{\left[\sum_{j=0}^{\infty}\alpha^j\mathbf{X}_{(n)}^j\right]}_{\equiv\mathbf{X}_{p,(n)}}\vec{x}_t
$$

$$
\Rightarrow \mathbf{H}_{(n)} = (1-\alpha)\mathbf{X}_{p,(n)}'\mathbf{H}_q
$$

We iterate until convergence of $\mathbf{H}_{(n)}$.

2.2.2 Implementation

The following instructions implement this model in DRIPs.

Include the package:

```
using DRIPs;
```

Assign values:

```
ρ   = 0.6;         #persistence of money growth
σ_u = 0.1;         #std. deviation of shocks to money growth
α   = 0.8;         #degree of strategic complementarity
L   = 40;          #length of truncation
Hq  = ρ.^(0:L-1); #state-space rep. of Δq
```

Primitives of the DRIP:

```
using LinearAlgebra;
ω   = 0.2;
β   = 0.99;
A   = [1 zeros(1,L-2) 0; Matrix(I,L-1,L-1) zeros(L-1,1)];
Q   = [σ_u; zeros(L-1,1)];
```

A Function for Finding the Fixed Point Let us now define a function that solves the GE problem and returns the solution in a Drip structure:

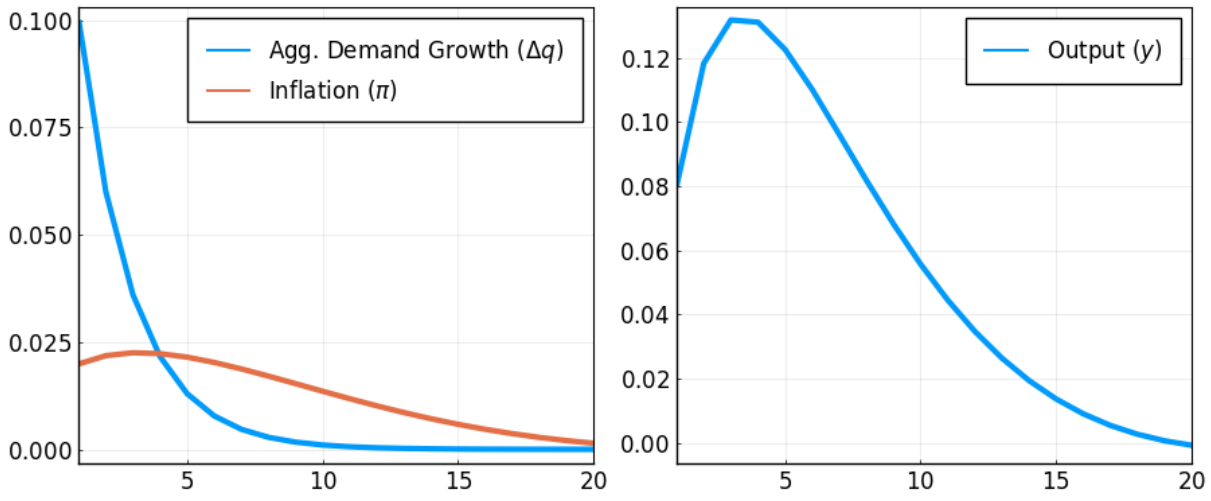
```
function ge_drip(ω,β,A,Q,   #primitives of drip
                 α,         #strategic complementarity
                 Hq,        #state space rep. of Δq
                 L;         #length of truncation
                 H0=Hq,     #optional: initial guess for H
                 maxit=200, #optional: max number of iterations for GE code
                 tol=1e-4)  #optional: tolerance for iterations
    err   = 1;
    iter  = 0;
    M     = [zeros(1,L-1) 0; Matrix(I,L-1,L-1) zeros(L-1,1)];
    while (err > tol) & (iter < maxit)
    if iter == 0
        global ge  = Drip(ω,β,A,Q,H0, w = 0.9);
    else
        global ge  = Drip(ω,β,A,Q,H0;Ω0 = ge.ss.Ω ,Σ0 = ge.ss.Σ_1,maxit=15);
    end

    XFUN(jj) = ((I-ge.ss.K*ge.ss.Y')*ge.A)^jj * (ge.ss.K*ge.ss.Y') * (M')^jj
    X = DRIPs.infinitesum(XFUN; maxit=L, start = 0);

    XpFUN(jj) = α^jj * X^(jj)
    Xp = DRIPs.infinitesum(XpFUN; maxit=L, start = 0);

    H1 = (1-α)*Xp'*Hq;
    err= 0.5*norm(H1-H0,2)/norm(H0)+0.5*err;
    H0 = H1;

    iter += 1;
    if iter == maxit
        print("GE loop hit maxit\n")
    end
    println("Iteration $iter. Difference: $err")
    end
    return(ge)
end;
```

Solution Solve for benchmark parameterization:

```
ge = ge_drip(ω,β,A,Q,α,Hq,L);
```

IRFs Get IRFs:

```
geirfs = irfs(ge,T = L)

M  = [zeros(1,L-1) 0; Matrix(I,L-1,L-1) zeros(L-1,1)]; #shift matrix
dq = diagm(Hq)*geirfs.x[1,1,:];                        #change in nominal demand
Pi = (I-M)*geirfs.a[1,1,:];                            #inflation
y  = inv(I-M)*(dq-Pi);                                 #output

using Plots, LaTeXStrings; pyplot();
p1 = plot(1:L,[dq,Pi],
label = [L"Agg. Demand Growth ($\Delta q$)" L"Inflation ($\pi$)"]);

p2 = plot(1:L,y,
label  = L"Output ($y$)");

plot(p1,p2,
layout     = (1,2),
xlim       = (1,20),
lw         = 3,
legend     = :topright,
legendfont = font(12),
tickfont   = font(12),
size       = (900,370),
framestyle = :box)
```



2.3 Replication of Maćkowiak and Wiederholt (2009)

This example replicates Maćkowiak and Wiederholt (2009) (henceforth MW) using the DRIPs package.

2.3.1 Setup

The problem in MW is

$$\begin{aligned}
&\min_{\{\hat{\Delta}_{i,t},\hat{z}_{i,t}\}} \left\{ E\left[(\Delta_t - \hat{\Delta}_{i,t})^2\right] + \underbrace{\left(\frac{\hat{\pi}_{14}}{\hat{\pi}_{11}}\right)^2}_{\equiv \xi} E\left[(z_{i,t} - \hat{z}_{i,t})^2\right] \right\}, \\
s.t.\quad & \mathcal{I}(\{\Delta_t\};\{\hat{\Delta}_{i,t}\}) + \mathcal{I}(\{z_{i,t}\};\{\hat{z}_{i,t}\}) \leq \kappa, \\
& \{\Delta_t, \hat{\Delta}_{i,t}\} \perp \{z_{i,t}, \hat{z}_{i,t}\}
\end{aligned}$$

where

$$\begin{aligned}
\Delta_t &\equiv p_t + \underbrace{\left(\frac{|\hat{\pi}_{13}|}{|\hat{\pi}_{11}|}\right)}_{\equiv 1-\alpha}(q_t - p_t) \\
p_t &= \int_0^1 \hat{\Delta}_{i,t} di \\
q_t &= \rho q_{t-1} + \nu_t, \nu_{q,t} \sim \mathcal{N}(0, \sigma_q^2) \\
z_{i,t} &= \rho z_{i,t-1} + \nu_{z,t}, \nu_{z,t} \sim \mathcal{N}(0, \sigma_z^2)
\end{aligned}$$

2.3.2 Mapping the Problem to a DRIP

There are a few ways of translating the problem above to a `Drip` structure; however, the most efficient way, due to the independence assumption, is to write it as the sum of two DRIPs: one that solves the attention problem for the idiosyncratic shock, and one that solves the attention problem for the aggregate shock which also has endogenous feedback.

Moreover, since the problem above has a fixed capacity, instead of a fixed cost of attention (ω) as in DRIPs pacakge, we need to iterate over ω's to find the one that corresponds with κ.

2.3.3 Initialization

Include the package:

```
using DRIPs;
```

Assign parameters:

```
ρ  = 0.95;
σq = 0.01;
σz = 11.8*σq;
κ  = 3;
ξ  = 1;
α  = 1 - 0.15;
```

Primitives of Drip:

```
using LinearAlgebra;
L  = 21; # length of trunction
# MW truncate the state space with linear irfs of length 20
A  = [zeros(1,L);[Matrix(I,L-1,L-1);zeros(1,L-1)]'];
Qq = zeros(L,1); Qq[1]=σq;
Qz = zeros(L,1); Qz[1]=σz;
H  = zeros(L,1); H[1:21] = Array(1:-1/20:0);
```

2.3.4 Functions

We start with a function that solves the aggregate problem with feedback for a given ω.

Solving for the fixed point given ω:

```
function agg_drip(ω,A,Qq,        #primitives of drip except for H
                  α,             #strategic complementarity
                  H;             #state space rep. of q
                  β     = 1,     #optional: discount factor
                  H0    = H,     #optional: initial guess for HΔ
                  maxit = 10000, #optional: max number of iterations for GE code
                  tol   = 1e-4,  #optional: tolerance for iterations
                  w     = 1)     #optional: update weight for RI
    errmin= 1;
    err   = 1;
    iter  = 0;
    L     = length(H);
    while (err > tol) & (iter < maxit)
        if iter == 0
            global agg = Drip(ω,β,A,Qq,H0;w=w);
        else
            global agg = Drip(ω,β,A,Qq,H0;Ω0=agg.ss.Ω,Σ0=agg.ss.Σ_1,w=w);
        end
        XFUN(jj) = ((I-agg.ss.K*agg.ss.Y')*agg.A)^jj *
                    (agg.ss.K*agg.ss.Y') * (agg.A')^jj
        X = DRIPs.infinitesum(XFUN; maxit=200, start = 0);
        XpFUN(jj) = α^jj * X^(jj)
        Xp = DRIPs.infinitesum(XpFUN; maxit=200, start = 0);
        H1 = (1-α)*Xp'*H;
        err= 0.5*norm(H1-H0,2)/norm(H0)+0.5*err;
        # perturb the initial guess if solution is the zero capacity one
        if DRIPs.capacity(agg) < 1e-2
            H0 = H0+rand(L).*(H-H0);
        else # store the solution if it has positive capacity
            H0 = H1;
            if err < errmin
                global aggmin = agg;
                errmin = err;
            end
        end
        iter += 1;
    end
    return(aggmin, errmin)
end;
```

Now we need a function that iterates over ω's to find the one that corresponds to a given capacity for the MW problem.

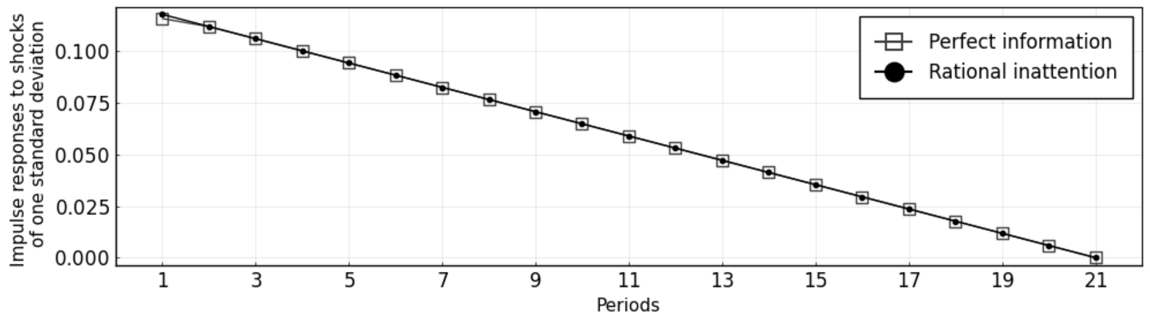
Solving for the optimal ω:

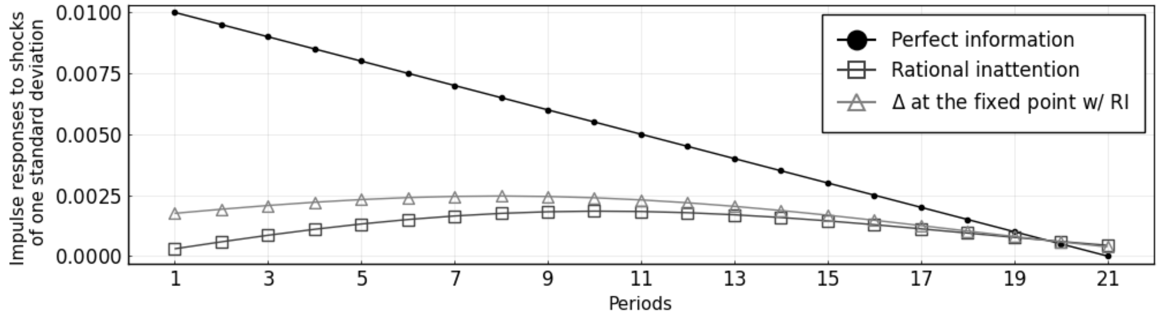
```
using Printf;
function MW(κ,α,A,Qq,Qz,Hq,Hz; #primitives of MW problem
            ω     = σq^2,      #optional: initial guess for ω
            tol   = 1e-3,      #optional: tolerance for κ
            maxit = 10000)     #optional: max iterations
    ωs    = [ω; 2*ω];
    caps  = [];
    iter  = 0;
    err   = 1;
    it    = 0;
    while (err > tol) & (iter < maxit)
        agg, errtemp = agg_drip(ω,A,Qq,α,H; H0 = rand(L),maxit=20,w=0.95);
        idi = Drip(ω,1,A,Qz,H,w = 0.9) ;
        cap = DRIPs.capacity(agg, unit="bit")+DRIPs.capacity(idi, unit="bit");
        x = ω/σq^2;
        @printf("ω = %.2fσq² for κ = %.2f \n",x,cap)
        push!(caps,cap);
        if it == 0
            ω = ωs[end];
        else
            slope = (caps[end]-caps[end-1])/(ωs[end]-ωs[end-1]);
            ω     = ω + (κ-caps[end])/slope;
            push!(ωs,ω);
        end
        err = abs(caps[end] - κ)/κ;
        it  += 1;
    end
    return(ω);
end;
```

2.3.5 Figures

Start with the benchmark calibration:

```
ω = MW(3,α,A,Qq,Qz,H,H);
agg, err  = agg_drip(ω,A,Qq,α,H; H0 = rand(L), maxit = 500, w = 0.95);
idi       = Drip(ω,1,A,Qz,H,w = 0.9);
@printf("Agg. Capacity = %.2f bits, Idio. Capacity = %.2f bits",
        DRIPs.capacity(agg),DRIPs.capacity(idi));
iirfs  = irfs(idi, T = L)
airfs  = irfs(agg, T = L)

using Plots, LaTeXStrings; pyplot();
p1 = plot([iirfs.a[1,1,:],σz*H],
label             = ["Perfect information" "Rational inattention"],
marker            = [:square :circle],
color             = [:gray25 :black],
markercolor       = [false :black],
markerstrokecolor = [:gray25 :black],
markersize        = [7 3],
xlabel            = "Periods",
ylabel            = "Impulse responses to shocks \n of one standard deviation");
p2 = plot([σq*H,airfs.a[1,1,1:end],σq*agg.H],
label             = ["Perfect information" "Rational inattention"
L"$\Delta$ at the fixed point w/ RI"],
marker            = [:circle :square :utriangle],
color             = [:black :gray25 :gray50],
markercolor       = [:black false false] ,
markerstrokecolor = [:black :gray25 :gray50],
markersize        = [3 7 7],
xlabel            = "Periods",
ylabel            = "Impulse responses to shocks \n of one standard deviation")

plot(p1,p2,
layout     = (2,1),
xlim       = (0,L+1),
lw         = 1,
legend     = :topright,
legendfont = font(12),
tickfont   = font(12),
size       = (1000,550),
xticks     = 1:2:21,
framestyle = :box)
```





For $\alpha = 0.7$:

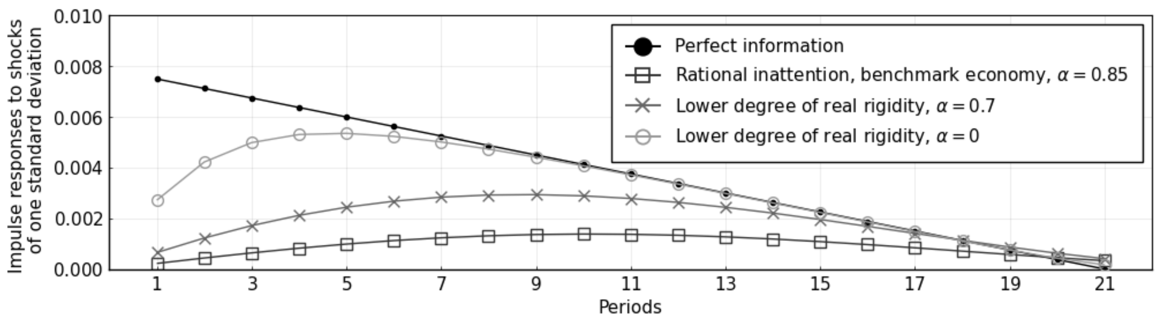
```
ω_α7 = MW(3,0.7,A,Qq,Qz,H,H);

agg_α7, err = agg_drip(ω_α7,A,Qq,0.7,H; H0 = rand(L), maxit = 100, w = 0.95);
idi_α7      = Drip(ω_α7,1,A,Qz,H,w = 0.9);

@printf("Agg. Capacity = %.2f bits, Idio. Capacity = %.2f bits",
DRIPs.capacity(agg_α7),DRIPs.capacity(idi_α7));
```

For $\alpha = 0$:

```
ω_α0 = MW(3,0,A,Qq,Qz,H,H);

agg_α0, err = agg_drip(ω_α0,A,Qq,0,H; H0 = rand(L), maxit = 100, w = 0.95);
idi_α0      = Drip(ω_α0,1,A,Qz,H,w = 0.9);

@printf("Agg. Capacity = %.2f bits, Idio. Capacity = %.2f bits",
DRIPs.capacity(agg_α0),DRIPs.capacity(idi_α0));
```

Plot IRFs:

```
airfs_α7  = irfs(agg_α7, T = L);
airfs_α0  = irfs(agg_α0, T = L);

plot(1:L,0.75*[σq*H,airfs.a[1,1,:],airfs_α7.a[1,1,:],airfs_α0.a[1,1,:]],
label              = ["Perfect information"
                      L"Rational inattention, benchmark economy, $\alpha = 0.85$"
                      L"Lower degree of real rigidity, $\alpha = 0.7$"
                      L"Lower degree of real rigidity, $\alpha = 0$"],
marker             = [:circle :square :x :circle],
color              = [:black :gray20 :gray40 :gray60],
markercolor        = [:black false :gray40 false],
markerstrokecolor = [:black :gray20 :gray40 :gray60],
markersize         = [3 7 7 7],
xlim               = (0,L+1),
lw                 = 1,
xticks             = 1:2:21,
legend             = :topright,
legendfont         = font(11),
tickfont           = font(11),
size               = (1000,275),
ylim               = (0,σq),
framestyle         = :box,
xlabel             = "Periods",
ylabel             = "Impulse responses to shocks \n of one standard deviation")
```



For $\kappa = 4$:

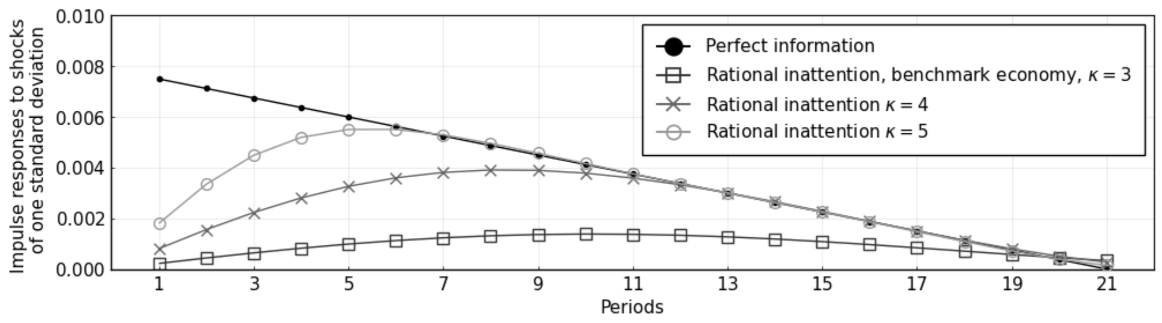
```
ω_κ4 = MW(4,α,A,Qq,Qz,H,H);

agg_κ4, err = agg_drip(ω_κ4,A,Qq,α,H; H0 = rand(L), maxit = 500, w = 0.95)
idi_κ4      = Drip(ω_κ4,1,A,Qz,H,w = 0.9)

@printf("Agg. Capacity = %.2f bits, Idio. Capacity = %.2f bits",
DRIPs.capacity(agg_κ4),DRIPs.capacity(idi_κ4));
```

For $\kappa = 5$:

```
ω_κ5 = MW(5,α,A,Qq,Qz,H,H; ω = 0.1*σq^2);

agg_κ5, err = agg_drip(ω_κ5,A,Qq,α,H; H0 = rand(L), maxit = 500, w = 0.95)
idi_κ5      = Drip(ω_κ5,1,A,Qz,H,w = 0.9)

@printf("Agg. Capacity = %.2f bits, Idio. Capacity = %.2f bits",
DRIPs.capacity(agg_κ5),DRIPs.capacity(idi_κ5));
```

Plot IRFs:

```
airfs_κ4  = irfs(agg_κ4, T = L);
airfs_κ5  = irfs(agg_κ5, T = L);

plot(1:L,0.75*[σq*H,airfs.a[1,1,:],airfs_κ4.a[1,1,:],airfs_κ5.a[1,1,:]],
label              = ["Perfect information"
                       L"Rational inattention, benchmark economy, $\kappa = 3$"
                       L"Rational inattention $\kappa = 4$"
                       L"Rational inattention $\kappa = 5$"],
marker             = [:circle :square :x :circle],
color              = [:black :gray20 :gray40 :gray60],
markercolor        = [:black false :gray40 false],
markerstrokecolor  = [:black :gray20 :gray40 :gray60],
markersize         = [3 7 7 7],
xlim               = (0,L+1),
lw                 = 1,
xticks             = 1:2:21,
legend             = :topright,
legendfont         = font(11),
tickfont           = font(11),
size               = (1000,275),
ylim               = (0,σq),
framestyle         = :box,
xlabel             = "Periods",
ylabel             = "Impulse responses to shocks \n of one standard deviation")
```



2.4 Replication of Sims (2010)

This example replicates Sims (2010) from the Handbook of Monetary Economics using the `DRIPs` package.

2.4.1 Setup

The problem in Sims (2010), as it appears on page 21, with slight change of notation,

$$
\begin{aligned}
&\min_{\{\Sigma_{t|t}\succeq 0\}_{t\geq 0}} \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t \left(tr(\Sigma_{t|t}\mathbf{H}\mathbf{H}') + \omega \log\left(\frac{|\Sigma_{t|t-1}|}{|\Sigma_{t|t}|}\right)\right)\right] \\
s.t. \quad & \Sigma_{t+1|t} = \mathbf{A}\Sigma_{t|t}\mathbf{A}' + \mathbf{Q}\mathbf{Q}' \\
& \Sigma_{t|t-1} - \Sigma_{t|t} \text{ positive semi-definite}
\end{aligned}
$$

where

$$
\mathbf{H} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 0.95 & 0 \\ 0 & 0.4 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} \sqrt{0.0975} & 0 \\ 0 & \sqrt{0.86} \end{bmatrix}
$$

We have renamed the parameters so that the problem directly maps to a `DRIP` Otherwise, the problem is the same.

2.4.2 Initialization

Include the package:

```
using DRIPs;
```

Set parameters:

```
β = 0.9;
ω = 1.0;
A = [0.95 0.0; 0.0 0.4];
Q = [0.0975^0.5 0.0; 0.0 0.86^0.5];
H = [1.0; 1.0];
```

2.4.3 Solution

Benchmark Parameterization Solve and display the optimal posterior covariance matrix:

```
sol_bp = Drip(ω,β,A,Q,H);
sol_bp.ss.Σ_p
```

```
2×2 Array{Float64,2}:
0.359213  -0.177025
-0.177025   0.794584
```

Lower Cost of Attention: $\omega = 0.1$ Solve and display the optimal posterior covariance matrix:

```
sol_lω = Drip(0.1,β,A,Q,H);
sol_lω.ss.Σ_p
```

```
2×2 Array{Float64,2}:
0.319919  -0.304142
-0.304142   0.386163
```

Different Discount Factors: $\beta \in 0, 1$ Solve the model for $\beta = 0$ and $\beta = 1$ to compare with the benchmark value of $\beta = 0.9$:

$\beta = 0$

```
sol_lβ = Drip(ω,0,A,Q,H);
sol_lβ.ss.Σ_p
```

```
2×2 Array{Float64,2}:
0.495403  -0.152171
-0.152171   0.808939
```

$\beta = 1$

```
sol_hβ = Drip(ω,1,A,Q,H);
sol_hβ.ss.Σ_p
```

```
2×2 Array{Float64,2}:
0.337666  -0.178019
-0.178019   0.799701
```

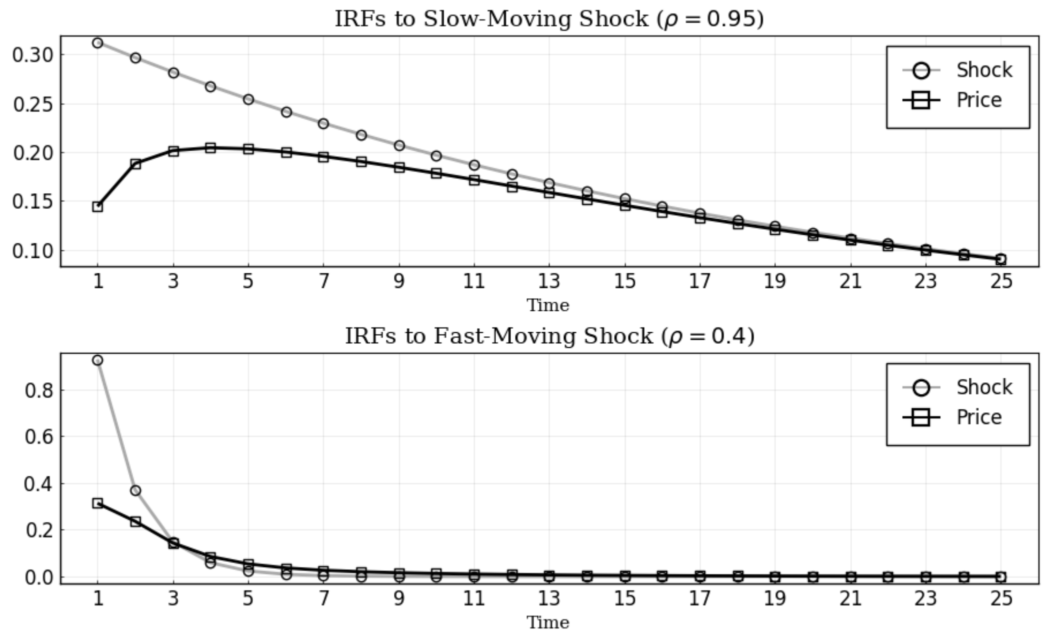
2.4.4 IRFs

Benchmark Parameterization Get the IRFs:

```
T = 25;
irfs_bp = irfs(sol_bp,T = T);
```

Plot the IRFs:

```
using Plots, LaTeXStrings; pyplot();
p1 = plot(1:T, [irfs_bp.x[1,1,:], irfs_bp.a[1,1,:]],
     title             = L"IRFs to Slow-Moving Shock ($\rho = 0.95$)",
     label             = ["Shock" "Price"],
     color             = [:darkgray :black],
     marker            = [:circle :square],
     markerstrokecolor = :match,
     markercolor       = false, markersize = 6)
p2 = plot(1:T, [irfs_bp.x[2,2,:], irfs_bp.a[1,2,:]],
     title             = L"IRFs to Fast-Moving Shock ($\rho = 0.4$)",
     label             = ["Shock" "Price"],
     color             = [:darkgray :black],
     marker            = [:circle :square],
     markerstrokecolor = :match,
     markercolor       = false, markersize = 6)
p = plot(p1,p2, layout = (2,1),
    xlabel     = "Time", lw = 2,
    xticks     = (1:2:T), xlim = (0,T+1),
    fontfamily = "serif",
    legend     = :topright,
    legendfont = font(12), tickfont = font(12),
    size       = (900,550),
    framestyle = :box)
```

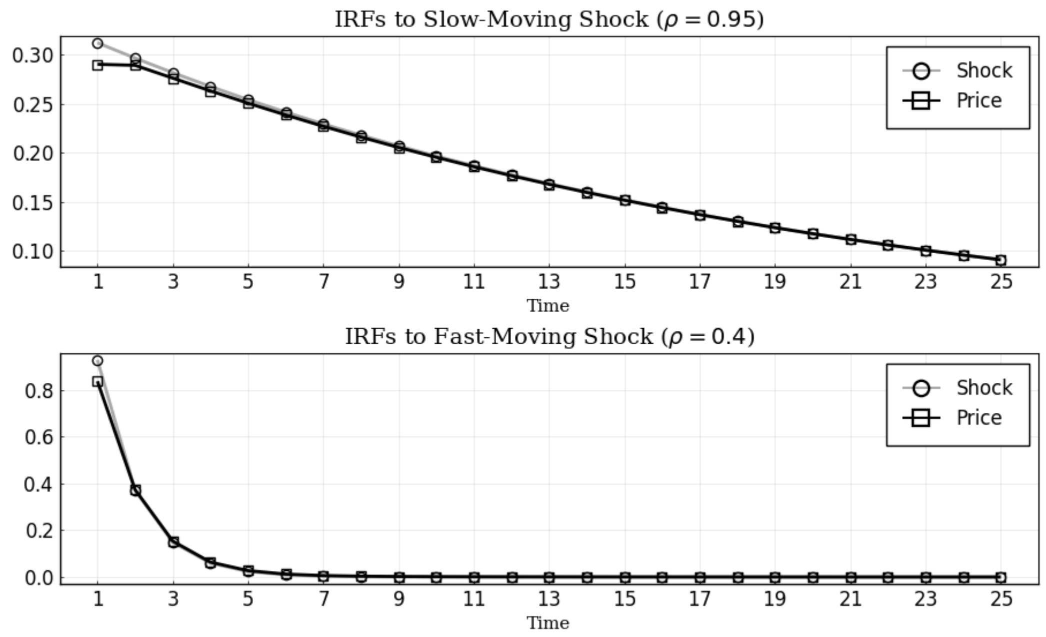


Lower Cost of Attention ($\omega = 0.1$) Get the IRFs:

```
T = 25; #length of IRFs
irfs_lω = irfs(sol_lω,T = T);
```

Plot the IRFs:

```
p1 = plot(1:T, [irfs_lω.x[1,1,:], irfs_lω.a[1,1,:]],
     title             = L"IRFs to Slow-Moving Shock ($\rho = 0.95$)",
     label             = ["Shock" "Price"],
     color             = [:darkgray :black], marker = [:circle :square],
     markerstrokecolor = :match, markercolor = false, markersize = 6)
p2 = plot(1:T, [irfs_lω.x[2,2,:], irfs_lω.a[1,2,:]],
     title             = L"IRFs to Fast-Moving Shock ($\rho = 0.4$)",
     label             = ["Shock" "Price"],
     color             = [:darkgray :black], marker = [:circle :square],
     markerstrokecolor = :match, markercolor = false, markersize = 6)
p = plot(p1,p2, layout = (2,1),
    xlabel     = "Time", lw = 2,
    xticks     = (1:2:T), xlim = (0,T+1),
    fontfamily = "serif", legend = :topright,
    legendfont = font(12), tickfont = font(12),
    size       = (900,550), framestyle = :box)
```

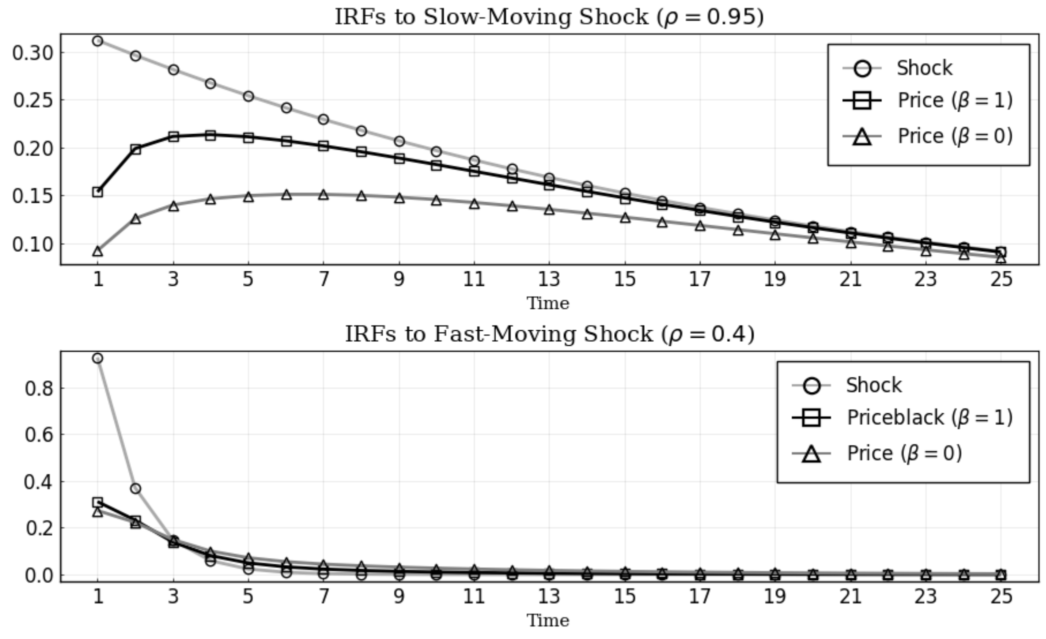


Other Discount Factors ($\beta \in 0, 1$) Get the IRFs:

```
T = 25; #length of IRFs
irfs_lβ = irfs(sol_lβ,T = T);
irfs_hβ = irfs(sol_hβ,T = T);
```

Plot the IRFs:

```
p1 = plot(1:T, [irfs_bp.x[1,1,:],irfs_hβ.a[1,1,:], irfs_lβ.a[1,1,:]],
     title       = L"IRFs to Slow-Moving Shock ($\rho = 0.95$)",
     label       = ["Shock" L"Price ($\beta=1$)" L"Price ($\beta=0$)"],
     color       = [:darkgray :black :gray50],
     marker      = [:circle :square :utriangle],
     markercolor = false, markerstrokecolor = :match, markersize = 6)
p2 = plot(1:T, [irfs_bp.x[2,2,:],irfs_hβ.a[1,2,:], irfs_lβ.a[1,2,:]],
     title       = L"IRFs to Fast-Moving Shock ($\rho = 0.4$)",
     label       = ["Shock" L"Priceblack ($\beta=1$)" L"Price ($\beta=0$)"],
     color       = [:darkgray :black :gray50],
     marker      = [:circle :square :utriangle],
     markercolor = false, markerstrokecolor = :match, markersize = 6)
p = plot(p1,p2, layout = (2,1), xlabel = "Time", lw = 2,
    xticks      = (1:2:T), xlim = (0,T+1), fontfamily = "serif",
    legendfont = font(12), legend = :topright, tickfont = font(12),
    size        = (900,550), framestyle = :box)
```



2.4.5 Extensions

Transition Dynamics of Attention In this section, we solve for the transition dynamics of the optimal posterior covariance matrix starting from an initial prior that is different from the steady state prior.

For instance let us consider a case where the firm is at the steady state of the rational inattention problem at time 0, with prior covariance matrix $\bar{\Sigma}_{-1}$, and it receives a one time treatment with a perfectly informative signal about its optimal price:

$$s_0 = \mathbf{H}'\vec{x}_0$$

Solve for the transition dynamics The function `Trip` solves for the transition dynamics automatically given the initial signal. Start by initializing the initial signal:

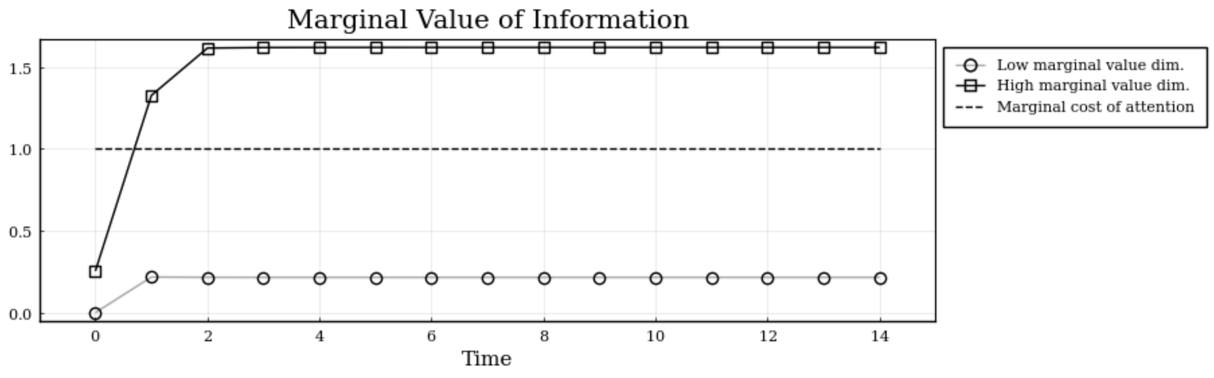
```
s0 = DRIPs.Signal(H,0.0);
```

Solve for the transition dynamics given s_0:

```
Tss     = 15; # guess for time until convergence
bp_trip = Trip(sol_bp, s0; T = Tss);
```

Plot Transition Path of Eigenvalues Plot the marginal values of information. In this problem the state is two dimensional. At any time, for every orthogonalized dimension, the agent weighs the marginal value of acquiring information in that dimension against the marginal cost of attention which is the parameter ω.The number of signals that the agent acquires at any time is the number of marginal values that are larger than ω.

```
p = plot(0:Tss-1,[bp_trip.Ds[1,1:Tss],
                  bp_trip.Ds[2,1:Tss],bp_trip.p.ω*ones(Tss,1)],
    label       = ["Low marginal value dim." "High marginal value dim."
                   "Marginal cost of attention"],
    title       = "Marginal Value of Information",
    size        = (900,275), xlabel = "Time",
    color       = [:darkgray :black :black], line = [:solid :solid :dash],
    marker      = [:circle :square :none], markercolor = false,
    markersize = 6, markerstrokecolor = :match,
    xlim        = (-1,Tss), xticks = 0:2:Tss-1,
    legend      = :outertopright, fontfamily = "serif",
    framestyle = :box)
```

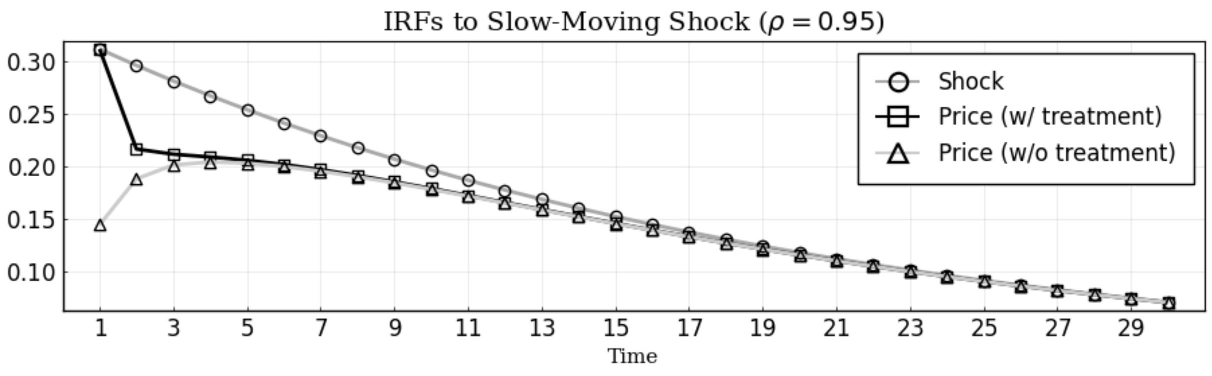


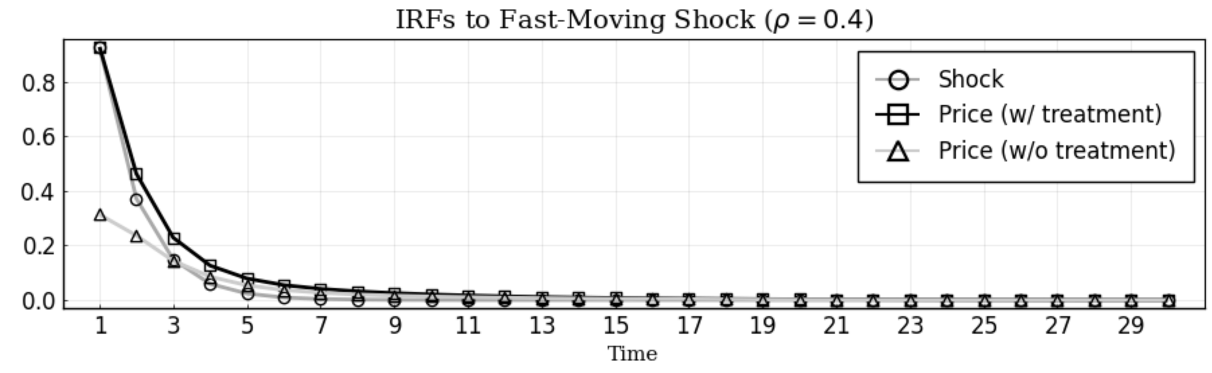
Impulse Response Functions with Information Treatment Get the IRFs in the transition path after treatment:

```
T = 30;
tirfs_bp = irfs(sol_bp,s0,T = T); # irfs with treatment
irfs_bp  = irfs(sol_bp,T = T);    # irfs in the Ss (without treatment)
```

Plot IRFs:

```
p1 = plot(1:T, [irfs_bp.x[1,1,:], tirfs_bp.a[1,1,:], irfs_bp.a[1,1,:]],
     title       = L"IRFs to Slow-Moving Shock ($\rho = 0.95$)",
     label       = ["Shock" "Price (w/ treatment)" "Price (w/o treatment)"],
     color       = [:darkgray :black :gray80],
     marker      = [:circle :square :utriangle],
     markercolor = false, markerstrokecolor = :match, markersize = 6)
p2 = plot(1:T, [tirfs_bp.x[2,2,:], tirfs_bp.a[1,2,:], irfs_bp.a[1,2,:]],
     title       = L"IRFs to Fast-Moving Shock ($\rho = 0.4$)",
     label       = ["Shock" "Price (w/ treatment)" "Price (w/o treatment)"],
     color       = [:darkgray :black :gray80],
     marker      = [:circle :square :utriangle],
     markercolor = false, markerstrokecolor = :match, markersize = 6)
p = plot(p1,p2, layout = (2,1),
    xlabel     = "Time", lw = 2,
    xticks     = (1:2:T), xlim = (0,T+1),
    fontfamily = "serif",
    legend     = :topright, legendfont = font(12),
    tickfont   = font(12),size = (900,550),
    framestyle = :box)
```





2.5 Replication of Maćkowiak et al. (2018)

This example replicates Maćkowiak et al. (2018) from the Handbook of Monetary Economics using the `DRIPs` package.

2.5.1 Ex. 1: AR(2) Process

In this example, authors assume that the optimal action follows an AR(2) process,

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \theta_0 \varepsilon_t$$

Then, we can write a state-space form as:

$$\begin{bmatrix} x_t \\ x_{t-1} \end{bmatrix} = \underbrace{\begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} x_{t-1} \\ x_{t-2} \end{bmatrix} + \underbrace{\begin{bmatrix} \theta_0 \\ 0 \end{bmatrix}}_{\mathbf{Q}} \varepsilon_t$$

We now characterize the optimal signal, $S_t = h_1 X_t + h_2 X_{t-1} + \psi_t$, as a function of ϕ_2 and the capacity for information processing (κ).

Include the package:

```
using DRIPs, LinearAlgebra;
```

Assign value to deep parameters and define the structure of the problem:

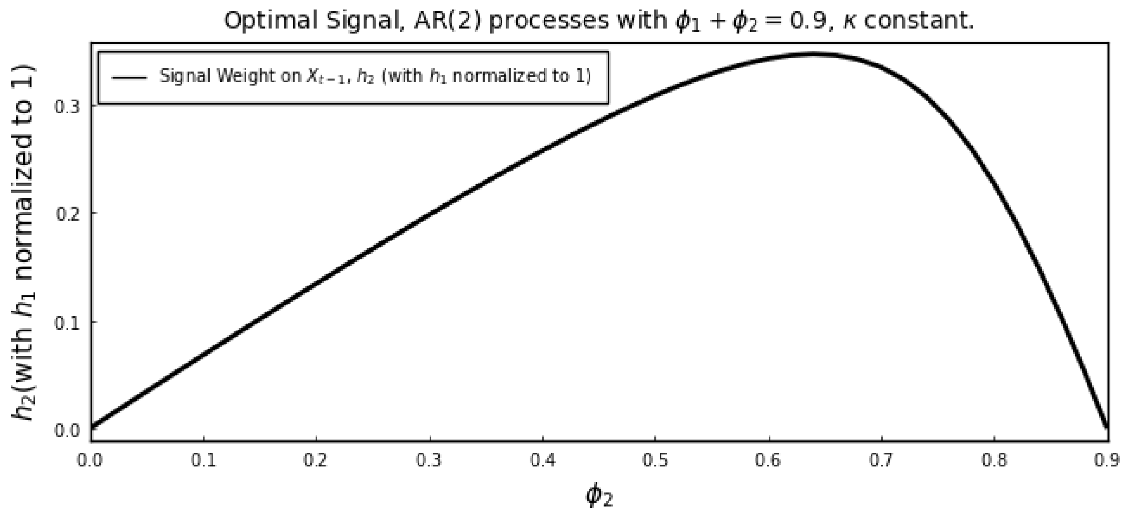
```
β       = 1.0 ; # Time preference
θ0      = 1.0 ;
p2_seq  = 0:0.02:0.9        ; # sequence of values for the AR(2) parameter
n_p     = length(p2_seq)    ;
κ_seq   = 0.008:0.01:3.5    ; # sequence of values for the information capacity
n_κ     = length(κ_seq)     ;
H       = [1; 0] ;
Q       = [θ0 ; 0] ;
```

Replication of Figures 2 and 3 Solve for different values of ϕ_2:

```
κ   = 0.2   ; # fix κ for this exercise so we can vary p2
h1  = zeros(n_p,1)    ;
h2  = zeros(n_p,1)    ;
h2norm = zeros(n_p,1) ;

for i in 1:n_p
    p2  = p2_seq[i] ;
    p1  = 0.9 - p2 ;
    A   = [p1 p2 ; 1.0 0.0] ;
    ex1a = Drip(κ,β,A,Q,H,fcap=true);
    h1[i] = ex1a.ss.Y[1]
    h2[i] = ex1a.ss.Y[2]
    h2norm[i] = ex1a.ss.Y[2]/ex1a.ss.Y[1] # normalize the first signal weight h1
end
```

Here is the replication of Figure 2 in Maćkowiak et al. (2018):

```
using Printf, LaTeXStrings, Plots; pyplot();
plot(p2_seq,h2norm[:,1],
     title       = L"Optimal Signal, AR(2) processes with
                     $\phi_1+\phi_2 = 0.9$, $\kappa$ constant.",
     xlabel      = L"$\phi_2$",
     ylabel      = L"$h_2$(with $h_1$ normalized to 1)",
     label       = L"Signal Weight on $X_{t-1}$, $h_2$ ($h_1$ normalized to 1)",
     legend      = :topleft, lw = 2, color = :black,
     xlim        = (0,0.9), xticks = (0:0.1:0.9),
     titlefont   = font(10), legendfont = font(7), tickfont = font(7),
     size        = (650,300), grid = :off, framestyle = :box)
```

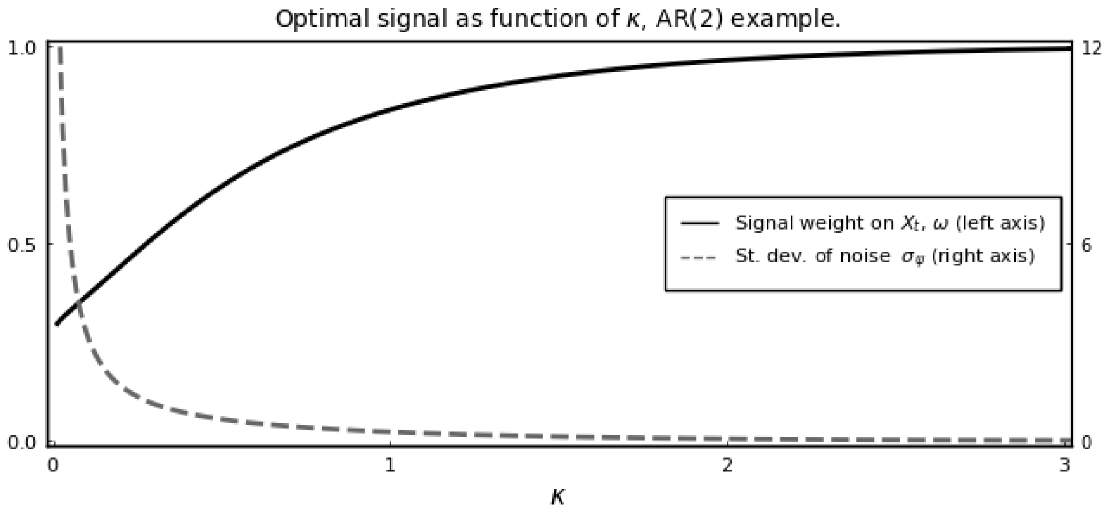


Now, we solve for the signal weight and standard deviation of noise, for different values of κ

```
p2     = 0.4          ;  # fix p2 in this exercise so we can vary κ
p1     = 0.99 - p2    ;
A      = [p1 p2 ; 1.0 0.0] ;
ω_sol  = zeros(n_κ,1) ;
σz_sol = zeros(n_κ,1) ;
for i in 1:n_κ
    κ = κ_seq[i] ;
    ex1b = Drip(κ,β,A,Q,H,fcap=true);
    h1temp = ex1b.ss.Y[1]
    h2temp = ex1b.ss.Y[2]
    ω_sol[i]  = 1.0 - (h2temp/h1temp)/(p2 + (1.0-p1)*(h2temp/h1temp)) ;
    ρ = (ω_sol[i] + (1.0-ω_sol[i])*p1)/h1temp
    σz_sol[i] = ρ*ex1b.ss.Σ_z[1,1] ;
end
```

Here is the replication of Figure 3 in Maćkowiak et al. (2018):

```
a = fill(NaN, n_κ, 1)
plot(κ_seq,[ω_sol[:,1] a],
     xlabel      = L"$\kappa$",
     label       = [L"Signal weight on $X_{t}$, $\omega$ (left axis)"
                    L"St. dev. of noise  $\sigma_{\psi}$ (right axis)"],
     title       = L"Optimal signal as function of $\kappa$, AR(2) example.",
     linestyle   = [:solid :dash], color = [:black :gray40],
     xticks      = (0:1:3), xlim = (-0.02,3.02),
     yticks      = (0:0.5:1), ylim = (-0.01,1.01),
     lw          = 2, grid = :off, legend = :right,
     titlefont   = font(10), legendfont = font(8), tickfont = font(8),
     framestyle  = :box)
plot!(twinx(),κ_seq,σz_sol[:,1],
     linestyle   = :dash, color = :gray40,
     label       = "", xlim = (-0.02,3.02),
     xticks      = false, yticks = (0:6:12),
     ylim        = (-0.12,12.12),
     lw          = 2, tickfont = font(7),
     size        = (650,300), grid = :off, framestyle  = :box)
```



2.5.2 Ex. 2: AR(3) and ARMA(2,1) Processes

Here, we replicate the AR(3) and ARMA(2,1) examples in Mackowiak, Matejka and Wiederholt (2018). Using a similar method to the AR(2) case, we can write the law of motion of optimal actions as a state-space form.

The AR(3) Example:

```
β       = 1.0           ;
θ0      = 1.0           ;
κ       = 10.8          ;

p1      = 1.5           ;
p2      = -0.9          ;
p3      = 0.1           ;

A   = [p1 p2 p3 ; 1 0 0 ; 0 1 0] ;
Q   = [θ0 ; 0; 0]   ;
H   = [1  ; 0; 0]   ;

ex2a  = Drip(κ,β,A,Q,H;tol=1e-8);
```

To be consistent with MMW(2018), scale the signal vector so that the weight on the first element is one ($h_1 = 1$):

```
h1 = 1;
h2 = ex2a.ss.Y[2]/ex2a.ss.Y[1]
h3 = ex2a.ss.Y[3]/ex2a.ss.Y[1]
```

Print the weights:

```
s = @sprintf("  h1 = %5.3f, h2 = %5.3f, h3 = %5.3f", h1, h2, h3)  ;
println(s) ;
```

```
h1 = 1.000, h2 = -0.475, h3 = 0.053
```

Since we scaled the signal vector, we also need to adjust the noise in the signal accordingly:

```
AdjNoise   = 3.879
AdjPara_ex2a= AdjNoise*ex2a.ss.Y[1]
```

Calculate IRFs:

```
irf_ex2a = irfs(ex2a; T=30) ;
xirf_ex2a   = irf_ex2a.x[1,1,:]
xhatirf_ex2a= irf_ex2a.x_hat[1,1,:]
x     = zeros(3,30);
xhat_noise_ex2a = zeros(3,30);
for ii in 1:30
    if ii==1
        xhat_noise_ex2a[:,ii] = ex2a.ss.K;
    else
        xhat_noise_ex2a[:,ii] = A*xhat_noise_ex2a[:,ii-1] +
                                (ex2a.ss.K*ex2a.ss.Y')*(x[:,ii] -
                                A*xhat_noise_ex2a[:,ii-1]);
    end
end
xhat_noise_ex2a = xhat_noise_ex2a[1,:]*AdjPara_ex2a ;
```

The ARMA(2,1) Example:

```
β       = 1.0          ;
θ0      = 0.5          ;
θ1      = -0.1         ;
κ       = 1.79         ;

p1      = 1.3          ;
p2      = -0.4         ;

A   = [p1 p2 θ1 ; 1.0 0.0 0.0 ; 0.0 0.0 0.0] ;
Q   = [θ0 ; 0.0; 1.0]   ;
H   = [1.0; 0.0; 0.0]   ;

ex2b = Drip(κ,β,A,Q,H,tol=1e-8);
```

To be consistent with MMW(2018), scale the signal vector so that the weight on the first element is one ($h_1 = 1$):

```
h1 = 1;
h2 = ex2b.ss.Y[2]/ex2b.ss.Y[1]
h3 = ex2b.ss.Y[3]/ex2b.ss.Y[1]
```

Print the weights:

```
s = @sprintf("  h1 = %5.3f, h2 = %5.3f, h3 = %5.3f", h1, h2, h3)  ;
println(s) ;
```

```
h1 = 1.000, h2 = -0.275, h3 = -0.069
```

Since we scaled the signal vector, we also need to adjust the noise in the signal accordingly:

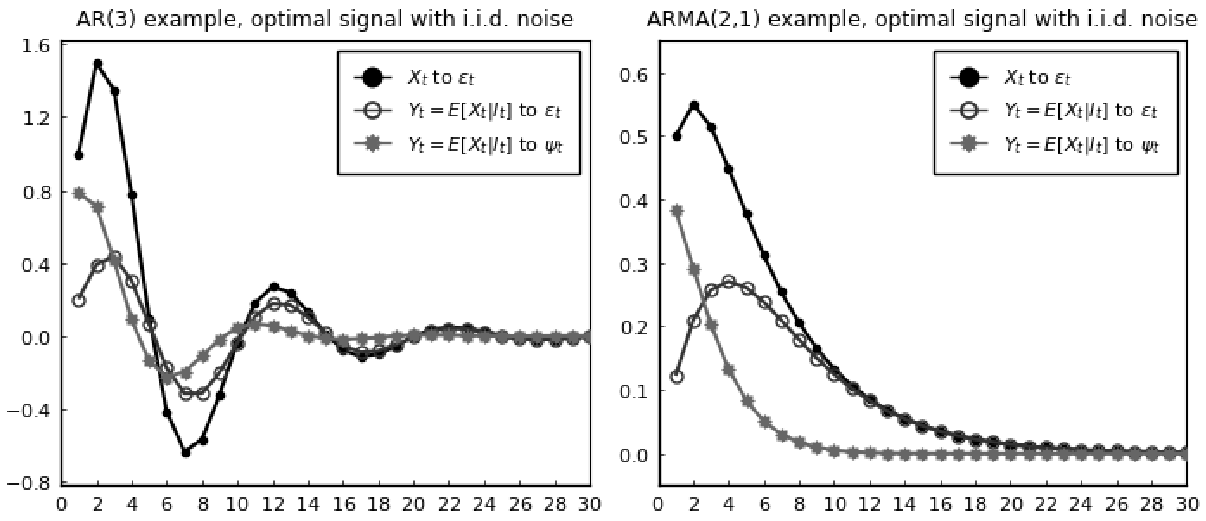
```
AdjNoise    = 1.349
AdjPara_ex2b= AdjNoise*ex2b.ss.Y[1]
```

Calculate IRFs:

```
irf_ex2b     = irfs(ex2b; T=30) ;
xirf_ex2b    = irf_ex2b.x[1,1,:]
xhatirf_ex2b = irf_ex2b.x_hat[1,1,:]
x     = zeros(3,30);
xhat_noise_ex2b = zeros(3,30);
for ii in 1:30
    if ii==1
        xhat_noise_ex2b[:,ii] = ex2b.ss.K;
    else
        xhat_noise_ex2b[:,ii] = A*xhat_noise_ex2b[:,ii-1] +
                                (ex2b.ss.K*ex2b.ss.Y')*(x[:,ii] -
                                A*xhat_noise_ex2b[:,ii-1]);
    end
end
xhat_noise_ex2b = xhat_noise_ex2b[1,:]*AdjPara_ex2b ;
```

Replication of Figure 5 in Maćkowiak et al. (2018):

```
p1 = plot(1:30,[xirf_ex2a xhatirf_ex2a xhat_noise_ex2a],
     title  = "AR(3) example, optimal signal with i.i.d. noise",
     color  = [:black :gray20 :gray40], ylim = (-0.82,1.62),
     yticks = (-0.8:0.4:1.6), markerstrokecolor = [:black :gray20 :gray40])
p2 = plot(1:30,[xirf_ex2b xhatirf_ex2b xhat_noise_ex2b],
     title  = "ARMA(2,1) example, optimal signal with i.i.d. noise",
     color  = [:black :gray20 :gray40],ylim = (-0.05,0.65),
     yticks = (0:0.1:0.6),  markerstrokecolor = [:black :gray20 :gray40])
Plots.plot(p1,p2, layout = (1,2),
     label  = [L"$X_{t}$ to $\varepsilon_t$"
               L"$Y_t=E[X_t|I_t]$ to $\varepsilon_t$"
               L"$Y_t=E[X_t|I_t]$ to $\psi_t$"],
     marker = [:circle :circle :star8], markercolor = [:black :false :gray40],
     legend = :topright, xticks = (0:2:30), markersize = [3 5 5],
     xlim   = (0,30), lw = 1.5, legendfont = font(8), guidefont=font(9),
     size   = (700,300), titlefont=font(9), tickfont=font(8),
     grid   = :off, framestyle = :box)
```



2.5.3 Ex. 3: Price-setting with Rational Inattention

We now replicate the price-setting exercise in Maćkowiak et al. (2018) and its comparison with the Woodford (2003) example. This corresponds to Figure (6) in their paper. The model structure is identifcal to our Example in Section 2.1 (without strategic complementarity) and Example in Section 2.2 (with strategic complementarity).

The Case with No Strategic Complementarity:

```
ρ   = 0.9;        #persistence of money growth
σ_u = 0.1;        #std. deviation of shocks to money growth
κ   = 0.62;
β   = 1.0;
A   = [1 ρ; 0 ρ];
Q   = σ_u*[1; 1];
H   = [1; 0];
ω   = 0.1;

ex_opt   = Drip(κ,β,A,Q,H,fcap=true);
capa_opt = DRIPs.capacity(ex_opt); #returns capacity utilized in bits
```

Calculate IRFs:

```
irfs_ex_opt = irfs(ex_opt, T = 12);
output_opt  = (irfs_ex_opt.x[1,1,:] - irfs_ex_opt.a[1,1,:]) ;
output_opt  = [0;output_opt] ;
```

Now, to compare with Woodford (2003), assume that firms observe a noisy signal, $S_t = q_t + \zeta_t$

where ζ_t is an idiosyncratic noise. We first define a function to solve the corresponding Kalman filtering problem.

```
function K_filtering(A,Q,Ysignal,Σz,Σ0 ; maxit=10000,tol=1e-10,w=1)
    err = 1
    iter = 0
    while (err > tol) & (iter < maxit)
        global Knew = Σ0*Ysignal*inv(Ysignal'*Σ0*Ysignal .+ Σz)
        global Σp_temp = Σ0 - Knew*Ysignal'*Σ0
        global Σ1 = A*Σp_temp*A' + Q*Q'

        err     = norm(Σ1 - Σ0,2)/norm(Σ0,2)
        Σ0 = w*Σ1 + (1-w)*Σ0

        #println("Iteration $iter. Difference: $err")
        iter += 1
    end
    return(Knew,Σ0,Σp_temp)
end;
```

Now find the capacity utilized under Woodford's formulation such that it yields the same information flow as the optimal signal under ratinoal inattention.

```
Ywoodford = [1;0];
Σ1_init   = ex_opt.ss.Σ_1;
Σz_new_b  = 0.01;
Σz_new_u  = 0.1;
Σz_new    = (Σz_new_b+Σz_new_u)/2;
for i in 1:10000
    (Knew,Σ1_new,Σp_temp) = K_filtering(A,Q,Ywoodford,Σz_new,Σ1_init;w=0.5);
    capa_woodford = 0.5*log(det(Σ1_new)/det(Σp_temp))/log(2);
    if capa_woodford > capa_opt
        global Σz_new_b  = Σz_new;
    else
        global Σz_new_u  = Σz_new;
    end
    global Σz_new    = (Σz_new_b+Σz_new_u)/2;
    err = abs(capa_woodford - capa_opt);
    if err < 1e-5
        break;
    end
end
```

Calculate impulse responses under Woodford's formulation

```
e_k = 1;
x   = zeros(2,12);
xhat= zeros(2,12);
a   = zeros(2,12);
for ii in 1:12
    if ii==1
        x[:,ii]     = Q*e_k;
        xhat[:,ii]  = (Knew*Ywoodford')*(x[:,ii]);
    else
        x[:,ii]     = A*x[:,ii-1];
        xhat[:,ii]  = A*xhat[:,ii-1]+(Knew*Ywoodford')*(x[:,ii]-A*xhat[:,ii-1]);
    end
    a[:,ii]  .= H'*xhat[:,ii];
end
output_woodford  = (x[1,:] - a[1,:]);
output_woodford  = [0;output_woodford];
```

The Case with Strategic Complementarity:

We now turn to the example with strategic complementarity. As in our Example in Section 2.2, we first define a function to solve the fixed point with endogenous feedback.

```
function ge_drip(ω,β,A,Q,           #primitives of drip except for H
                 α,                 #strategic complementarity
                 Hq,                #state space rep. of Δq
                 L;                 #length of truncation
                 H0       = Hq,     #optional: initial guess for H
                 maxit    = 200,    #optional: max number of iterations for GE
                 tol      = 1e-4)   #optional: tolerance for iterations
    err   = 1;
    iter  = 0;
    M     = [zeros(1,L-1) 0; Matrix(I,L-1,L-1) zeros(L-1,1)];
    while (err > tol) & (iter < maxit)
        if iter == 0
            global ge  = Drip(ω,β,A,Q,H0; w=0.9);
        else
            global ge  = Drip(ω,β,A,Q,H0; Ω0=ge.ss.Ω, Σ0=ge.ss.Σ_1, maxit=100);
        end
        XFUN(jj) = ((I-ge.ss.K*ge.ss.Y')*ge.A)^jj * (ge.ss.K*ge.ss.Y') * (M')^jj
        X = DRIPs.infinitesum(XFUN; maxit=L, start = 0);
        XpFUN(jj) = α^jj * X^(jj)
        Xp = DRIPs.infinitesum(XpFUN; maxit=L, start = 0);
        H1 = (1-α)*Xp'*Hq;
        err= 0.5*norm(H1-H0,2)/norm(H0)+0.5*err;
        H0 = H1;
        iter += 1;
        if iter == maxit
            print("GE loop hit maxit\n")
        elseif mod(iter,10) == 0
            println("Iteration $iter. Difference: $err")
        end
    end
    print(" Iteration Done.\n")
    return(ge)
end;
```

Now, we solve for the optimal signal structure under rational inattention.

```
ρ   = 0.9;         #persistence of money growth
σ_u = 0.1;         #std. deviation of shocks to money growth
α   = 0.85;         #degree of strategic complementarity
L   = 40;          #length of truncation
Hq  = ρ.^(0:L-1); #state-space rep. of Δq

ω   = 0.08;
β   = 1 ;
A   = [1 zeros(1,L-2) 0; Matrix(I,L-1,L-1) zeros(L-1,1)];
M   = [zeros(1,L-1) 0; Matrix(I,L-1,L-1) zeros(L-1,1)]; # shift matrix
Q   = [σ_u; zeros(L-1,1)];

ex_ge   = ge_drip(ω,β,A,Q,α,Hq,L) ;
capa_ge = DRIPs.capacity(ge) ;
```

Calculate IRFs:

```
geirfs  = irfs(ex_ge,T = L) ;

dq = diagm(Hq)*geirfs.x[1,1,:];
q  = inv(I-M)*dq ;
output_ge_opt  = q - geirfs.a[1,1,:] ;
output_ge_opt  = [0;output_ge_opt] ;
```

Finally, to compare with the IRFs under Woodford (2003)'s specification, we find signal noise such that it yields the same information flow as the optimal signal structure.

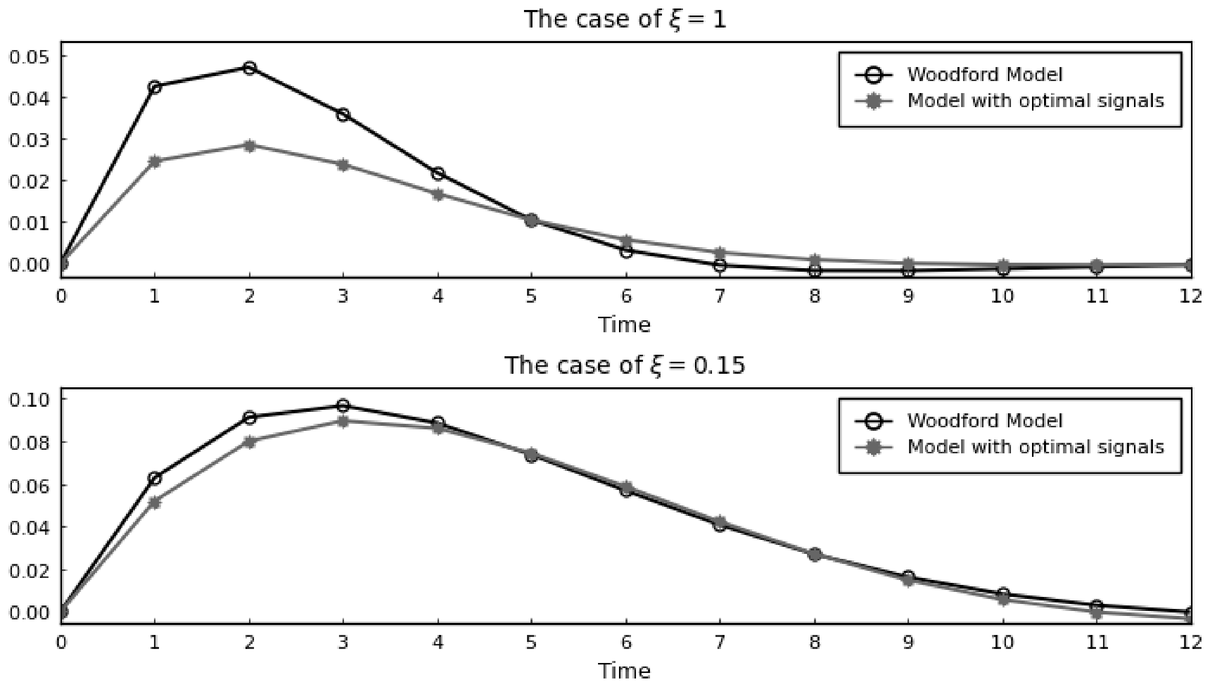
```
Ywoodford_ge = Hq
Σ1_init  = ex_ge.ss.Σ_1
Σz_new_b = 0.05
Σz_new_u = 0.12
Σz_new   = (Σz_new_b+Σz_new_u)/2
for i in 1:10000
    (Knew,Σ1_new,Σp_temp) = K_filtering(A,Q,Ywoodford_ge,Σz_new,Σ1_init;w=0.5)
    capa_woodford = 0.5*log(det(Σ1_new)/det(Σp_temp))/log(2)
    if capa_woodford > capa_ge
        global Σz_new_b  = Σz_new
    else
        global Σz_new_u  = Σz_new
    end
    global Σz_new    = (Σz_new_b+Σz_new_u)/2
    err = abs(capa_woodford - capa_ge)
    if err < 1e-5
        break
    end
end
XFUN(jj) = ((I-Knew*Ywoodford_ge')*A)^jj * (Knew*Ywoodford_ge') * (M')^jj
X = DRIPs.infinitesum(XFUN; maxit=L, start = 0);
XpFUN(jj) = α^jj * X^(jj)
Xp = DRIPs.infinitesum(XpFUN; maxit=L, start = 0);
H1 = (1-α)*Xp'*Hq;
```

Calculate impurse responses under Woodford's signal

```
e_k = 1;
x   = zeros(40,40);
xhat= zeros(40,40);
a   = zeros(1,40);
for ii in 1:40
    if ii==1
        x[:,ii]   = Q*e_k;
        xhat[:,ii]= (Knew*Ywoodford_ge')*(x[:,ii]);
    else
        x[:,ii]   = A*x[:,ii-1];
        xhat[:,ii]= A*xhat[:,ii-1]+(Knew*Ywoodford_ge')*(x[:,ii]-A*xhat[:,ii-1]);
    end
    a[:,ii] .= H1'*xhat[:,ii];
end
dq = diagm(Hq)*x[1,:];
q  = inv(I-M)*dq ;
output_ge_woodford = q - a[1,:] ;
output_ge_woodford = [0;output_ge_woodford] ;
```

We now have all the IRFs to replicate Figure 6 in Maćkowiak et al. (2018):

```
p1 = plot(0:12,[output_woodford output_opt],
     title       = L"The case of $\xi=1$",
     color       = [:black :gray40], markerstrokecolor = [:black :gray40],
     ylim        = (-0.003,0.053), ytick = (0:0.01:0.05))
p2 = plot(0:12,[output_ge_woodford[1:13] output_ge_opt[1:13]],
     title       = L"The case of $\xi=0.15$",
     color       = [:black :gray40], markerstrokecolor = [:black :gray40],
     ylim        = (-0.005,0.105), ytick = (0:0.02:0.1))
Plots.plot(p1,p2, layout = (2,1),
      label       = ["Woodford Model" "Model with optimal signals"],
      legend      = :topright,
      marker      = [:circle :star8], markersize = [5 5],
      markercolor = [:false :gray40],
      xlim        = (0,12), xtick = (0:1:12),
      xlabel      = "Time", lw = 1.5,
      legendfont  = font(8), titlefont = font(10), guidefont = font(9),
      tickfont    = font(8), size = (700,400),
      grid        = :off, framestyle = :box)
```



2.5.4 Ex. 4: Business Cycle Model with News Shocks

In this section, we replicate the business cycle model with news shocks in Section 7 in Maćkowiak et al. (2018).

Full-Information The techonology shock follows AR(1) process:

$$z_t = \rho z_{t-1} + \sigma \varepsilon_{t-k}$$

and the total labor input is:

$$n_t = \int_0^1 n_{i,t} di.$$

Under perfect information, the households chooses the utility-maximizing labor supply, all firms choose the profit-maximizing labor input, and the labor market clearing condition is:

$$\frac{1-\gamma}{\psi+\gamma} w_t = \frac{1}{\alpha}(z_t - w_t).$$

Then, the market clearing wages and the equilibrium labor input are:

$$w_t = \frac{\frac{1}{\alpha}}{\frac{1-\gamma}{\psi+\gamma} + \frac{1}{\alpha}} z_t \equiv \xi z_t$$
$$n_t = \frac{1}{\alpha}(1-\xi) z_t.$$

Rational Inattention Firms wants to keep track of their ideal price,

$$n_t^* = \frac{1}{\alpha} z_t - \frac{1}{\alpha} \frac{\psi+\gamma}{1-\gamma} n_t$$

where $n_t = \int_0^1 n_{i,t} di$. Then, firm i's choice depends on its information set at time t:

$$n_{i,t} = E_{i,t}[n_t^*].$$

Note that now the state space representation for n_t^* is determined in the equilibrium. However, we know that this is a Guassian process and by Wold's theorem we can decompose it to its $MA(\infty)$ representation:

$$n_t^* = \Phi(L)\varepsilon_t$$

where $\Phi(.)$ is a lag polynomial and ε_t is the shock to technology. Here, we have basically guessed that the process for $p_{i,t}^*$ is determined uniquely by the history of monetary shocks which requires that rational inattention errors of firms are orthogonal. Our objective is to find $\Phi(.)$.

Now, as in our Example in Section 2.2, we can represent the problem in a matrix notation.

Now, we solve for the optimal signal structure under rational inattention.

```
β   = 1     ;   # Time preference
γ   = 1/3   ;   # Inverse of intertemporal elasticity of substitution
ψ   = 0     ;   # Inverse of Frisch elasticity
α   = 3/4   ;   # Labor share in production function
θ   = -1/α*(ψ+γ)/(1-γ)  ;
ξ   = θ/(θ-1)   ;
ρ   = 0.9   ;   #persistence of technology shocks
σ   = 1     ;   #std. deviation of technology shocks
ω   = 6.5   ;   # Information cost
L   = 40    ;   #length of truncation
k   = 8     ;   #news horizon
M   = [zeros(1,L-1) 0; Matrix(I,L-1,L-1) zeros(L-1,1)]; # shift matrix
Hz  = ρ.^(0:L-1)
Hz  = (M^k)*Hz
A   = M     ;
Q   = [σ; zeros(L-1,1)] ;
```

Also, define a function that solves the GE problem and returns the solution in a Drip structure:

```
function ge_drip(ω,β,A,Q,α,θ,     #primitives of drip except for H
                 Hz,              #state space rep. of z
                 L;               #length of truncation
                 w_out    = 0.5,
                 H0       = Hz,   #optional: initial guess for H
                 maxit    = 200,  #optional: max number of iterations for GE
                 tol      = 1e-6) #optional: tolerance for iterations
    err   = 1;
    iter  = 0;
    M     = [zeros(1,L-1) 0; Matrix(I,L-1,L-1) zeros(L-1,1)];
    eye   = Matrix(I,L,L)
    while (err > tol) & (iter < maxit)
        if iter == 0
            global ge  = Drip(ω,β,A,Q,H0;w=0.5, tol=1e-8);
        else
            global ge  = Drip(ω,β,A,Q,H0;w=0.9, tol=1e-8,
                              Ω0=ge.ss.Ω, Σ0=ge.ss.Σ_1, maxit=1000);
        end
        XFUN(jj) = ((eye-ge.ss.K*ge.ss.Y')*ge.A)^jj*(ge.ss.K*ge.ss.Y') * (M')^jj
        X  = DRIPs.infinitesum(XFUN; maxit=L, start = 0);
        H1 = (1/α)*Hz + θ*X'*H0 ;
        err= 0.5*norm(H1-H0,2)/norm(H0)+0.5*err;
        H0 = w_out*H1 + (1.0-w_out)*H0 ;
        iter += 1;
        if iter == maxit
            print("GE loop hit maxit\n")
        elseif mod(iter,10) == 0
            println("Iteration $iter. Difference: $err")
        end
    end
    print(" Iteration Done.\n")
    return(ge)
end;
```

Solve the model:

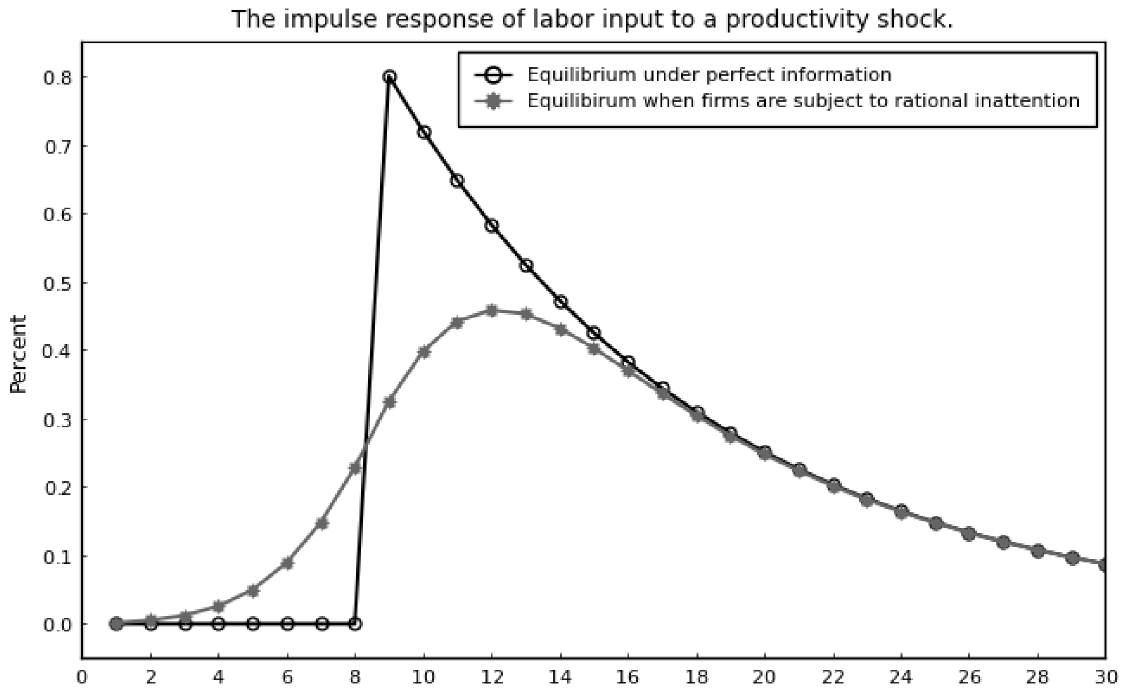
```
ge = ge_drip(ω,β,A,Q,α,θ,Hz,L)  ;
```

IRFs:

```
geirfs  = irfs(ge,T = L);
profit_loss = sum((geirfs.x[1,1,:]/100 - geirfs.x_hat[1,1,:]/100).^2);
s = @sprintf("==: Profit loss from rational inattention = %6.5f", profit_loss);
println(s) ;
```

Replication of Figure 7 in Maćkowiak et al. (2018)

```
n_opt   = geirfs.a[1,1,:]   ; # Optimal labor input under rational inattention
n_fullinfo = σ*1/α*(1-ξ)*Hz ; # Optimal labor input under full information
plot(1:30,[n_fullinfo[1:30] n_opt[1:30]],
     title      = "The impulse response of labor input to a productivity shock.",
     ylabel     = "Percent",
     label      = ["Equilibrium under perfect information"
                    "Equilibirum when firms are subject to rational inattention"],
     legend     = :topright, color = [:black :gray40],
     markerstrokecolor = [:black :gray40], markersize = [5 5],
     marker     = [:circle :star8], markercolor = [:false :gray40],
     ylim       = (-0.05,0.85), ytick = (0:0.1:0.8),
     xlim       = (0,30), xtick = (0:2:30), tickfont = font(8),
     lw         = 1.5, legendfont = font(8), titlefont = font(10),
     guidefont  = font(9), size = (650,400),
     grid       = :off, framestyle = :box)
```



2.6 Replication of the Quantitative Analysis in Afrouzi and Yang (2019)

This example replicates qunatitative analysis in Afrouzi and Yang (2019) using the `DRIPs` package.

2.6.1 Setup

Households Households are fully rational and maximize their life-time utilities:

$$\max \ \mathbb{E}_t^f \left[\sum_{t=0}^{\infty} \beta^t \left(\frac{C_t^{1-\sigma}}{1-\sigma} - \frac{\int_0^1 L_{i,t}^{1+\psi} di}{1+\psi} \right) \right]$$

$$\text{s.t.} \quad \int P_{i,t} C_{i,t} di + B_t \leq R_{t-1} B_{t-1} + \int_0^1 W_{i,t} L_{i,t} di + \Pi_t, \quad \text{for all } t$$

where

$$C_t = \left(\int C_{i,t}^{\frac{\theta-1}{\theta}} di \right)^{\frac{\theta}{\theta-1}}.$$

Here $\mathbb{E}t^f\left[\cdot\right]$ is the full information rational expectation operator at time t. Since the main purpose of this paper is to study the effects of nominal rigidity and rational inattention among firms, I assume that the household is fully informed about all prices and wages. Bt is the demand for nominal bond and R_{t-1} is the nominal interest rate. $L_{i,t}$ is firm-specific labor supply of the household, $W_{i,t}$ is the firm-specific nominal wage, and Π_t is the aggregate profit from the firms. C_t is the aggregator over the consumption for goods produced by firms. θ is the constant elasticity of substitution across different firms.

Firms There is a measure one of firms, indexed by i, that operate in monopolistically competitive markets. Firms take wages and demands for their goods as given, and choose their prices $P_{i,t}$ based on their information set, S_i^t, at that time. After setting their prices, firms hire labor from a competitive labor market and produce the realized level of demand that their prices induce with a production function,

$$Y_{i,t} = A_t L_{i,t},$$

where $L_{i,t}$ is firm i's demand for labor. I assume that shocks to A_t are independently and identically distributed and the log of the productivity shock, $a_{i,t} \equiv \log(A_t)$, follows a AR(1) process:

$$a_t = \rho_a a_{t-1} + \varepsilon_{a,t}, \ \ \varepsilon_{a,t} \sim N(0, \sigma_a^2).$$

Then, firm i's nominal profit from sales of all goods at prices $P_{i,j,t}{}_{j=1}^{N}$ is given by

$$\Pi_{i,t}(P_{i,t}, A_t, W_{i,t}, P_t, Y_t) = (P_{i,t} - W_{i,t}A_t)\left(\frac{P_{i,t}}{P_t}\right)^{-\theta} Y_t,$$

where Y_t is the nominal aggregate demand.

At each period, firms optimally decide their prices and signals subject to costs of processing information. Firms are rationally inattentive in a sense that they choose their optimal information set by taking into account the cost of obtaining and processing information. At the beginning of period t, firm i wakes up with its initial information set, S_i^{t-1}. Then it chooses optimal signals, $s_{i,t}$, from a set of available signals, $\mathcal{S}i, t$, subject to the cost of information which is linear in Shannon's mutual information function. Denote ω as the marginal cost of information processing. Firm i forms a new information set, $S_i^t = S_i^{t-1} \cup si, t$, and sets its new prices, $P_{i,t}$, based on that.

The firm i chooses a set of signals to observe over time $(s_{i,t} \in \mathcal{S}i, t)t = 0^\infty$ and a pricing strategy that maps the set of its prices at $t-1$ and its information set at t to its optimal price at any given period, $P_{i,t} : (S_i^t) \to \mathbb{R}$ where $S_i^t = S_i^{t-1} \cup s_{i,t} = S_i^{-1} \cup s_{i,\tau}{}_{\tau=0}^{t}$ is the firm's information set at time t. Then, the firm i's problem is to maximize the net present value of its life time profits given an initial information set:

$$\max_{\{s_{i,t}\in\mathcal{S}_{i,t}, P_{i,t}(S_i^t)\}_{t\geq 0}} \mathbb{E}\left[\sum_{t=0}^{\infty}\beta^t\Lambda_t\left\{\Pi_{i,t}(P_{i,t}, A_t, W_{i,t}, P_t, Y_t) - \omega\mathbb{I}(S_i^t; (A_\tau, W_{i,\tau}, P_\tau, Y_\tau)_{\tau\leq t}|S_i^{t-1})\right\}\middle| S_i^{-1}\right]$$
$$\text{s.t.} \quad S_i^t = S_i^{t-1} \cup s_{i,t}$$

where Λ_t is the stochastic discount factor and $\mathbb{I}(S_i^t; (A_\tau, W_{i,\tau}, P_\tau, Y_\tau)_{\tau\leq t}|S_i^{t-1})$ is the Shannon's mutual information function.

Monetary Policy Monetary policy is specified as a standard Talor rule:

$$R_t = (R_{t-1})^\rho \left(\Pi_t^{\phi_\pi}\left(\frac{Y_t}{Y_t^n}\right)^{\phi_x}\left(\frac{Y_t}{Y_{t-1}}\right)^{\phi_{\Delta y}}\right)^{1-\rho} \exp(u_t)$$

where $u_t \sim N(0, \sigma_u^2)$ is the monetary policy shock.

2.6.2 A Three-Equation GE Rational Inattention Model

Our general equlibrium model is characterized by the following three equations with two stochastic processes of technology (a_t) and monetary policy shocks (u_t):

$$\begin{aligned} x_t &= \mathbb{E}_t^f \left[x_{t+1} - \frac{1}{\sigma}\left(i_t - \pi_{t+1}\right)\right] + \mathbb{E}_t^f \left[y_{t+1}^n\right] - y_t^n \\ p_{i,t} &= \mathbb{E}_{i,t}\left[p_t + \alpha x_t\right] \\ i_t &= \rho i_{t-1} + (1-\rho)\left(\phi_\pi \pi_t + \phi_x x_t - \phi_{\Delta y}\Delta y_t\right) + u_t \end{aligned}$$

where $\mathbb{E}i,t[\cdot]$ is the firm i's expectation operator conditional on her time t information set, $x_t = y_t - y_t^n$ is the output gap, $yt^n = \frac{1+\psi}{\sigma+\psi}a_t$ is the natural level of output, i_t is the nominal interest rate, and $\alpha = \frac{\sigma+\psi}{1+\psi\theta}$ is the degree of strategic complementarity.

2.6.3 Matrix Representation

Firms wants to keep track of their ideal price, $p_{i,t}^* = p_t + \alpha x_t$. Notice that the state space representation for $p_{i,t}^*$ is no longer exogenous and is determined in the equilibrium. However, we know that this is a Guassian process and by Wold's theorem we can decompose it to its $MA(\infty)$ representation:

$$p_{i,t}^* = \Phi_a(L)\varepsilon_{a,t} + \Phi_u(L)\varepsilon_{u,t}$$

where $\Phi_a(.)$ and $\Phi_u(.)$ are lag polynomials. Here, we have basically guessed that the process for $p_{i,t}^*$ is determined uniquely by the history of monetary shocks which requires that rational inattention errors of firms are orthogonal.

Since we cannot put $MA(\infty)$ processes in the computer and have to truncate them. However, we know that for stationary processes we can arbitrarily get close to the true process by truncating $MA(\infty)$ processes. Our problem here is that $p_{i,t}^*$ has a unit root and is not stationary. We can bypass this issue by re-writing the state space in the following way:

$$p_{i,t}^* = \Phi_a(L)\varepsilon_{a,t} + \phi_u(L)\tilde{\varepsilon}_{u,t}, \quad \tilde{\varepsilon}_{u,t} = (1-L)^{-1}\varepsilon_{u,t} = \sum_{j=0}^{\infty}\varepsilon_{u,t-j}$$

here $\tilde{\varepsilon}_{u,t}$ is the unit root of the process and basically we have differenced out the unit root from the lag polynomial, and $\phi_u(L) = (1-L)\Phi_u(L)$. Notice that since the original process was difference stationary, differencing out the unit root means that $\phi_u(L)$ is now in ℓ_2, and the process can now be approximated arbitrarily precisely with truncation.

For ease of notation, let $z_t = (\varepsilon_{a,t}, \varepsilon_{u,t})$ and $\tilde{z}t = (\varepsilon a, t, \tilde{\varepsilon}_{u,t})$. For a length of truncation L, let $\vec{x}t' \equiv (zt, zt-1, \ldots, zt-(L+1)) \in \mathbb{R}^{2L}$ and $\vec{\mathbf{x}}t' \equiv (\tilde{z}t, \tilde{z}t-1, \ldots, \tilde{z}t-(L+1)) \in \mathbb{R}^{2L}$. Notice that

$$\vec{x}_t = (\mathbf{I} - \mathbf{\Lambda}\mathbf{M}')\vec{\mathbf{x}}_t$$
$$\vec{\mathbf{x}}_t = (\mathbf{I} - \mathbf{\Lambda}\mathbf{M}')^{-1}\vec{x}_t$$

where $\mathbf{I}$ is a 2×2 identity matrix, $\mathbf{\Lambda}$ is a diagonal matrix where $\mathbf{\Lambda}(2i, 2i) = 1$ and $\mathbf{\Lambda}(2i-1, 2i-1) = 0$ for all $i = 1, 2, \cdots, L$, and $\mathbf{M}$ is a shift matrix:

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 \end{bmatrix}$$

Then, note that $p^*_{i,t} \approx \mathbf{H}'\vec{\mathbf{x}}_t$ where $\mathbf{H} \in \mathbb{R}^{2L}$ is the truncated matrix analog of the lag polynominal, and is endogenous to the problem. Our objective is to find the general equilibrium $\mathbf{H}$ along with the optimal information structure that it implies.

Moreover, note that

$$a_t = \mathbf{H}'_a\vec{x}_t, \quad \mathbf{H}'_a = (1, 0, \rho_a, 0, \rho_a^2, 0, \ldots, \rho_a^{L-1}, 0)$$
$$u_t = \mathbf{H}'_u\vec{x}_t, \quad \mathbf{H}'_u = (0, 1, 0, 0, 0, 0, \ldots, 0, 0)$$

We will solve for $\mathbf{H}$ by iterating over the problem. In particular, in iteration $n \geq 1$, given the guess $\mathbf{H}_{(n-1)}$, we have the following state space representation for the firm's problem

$$\vec{\mathbf{x}}_t = \underbrace{\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \vec{\mathbf{x}}_{t-1} + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}}_{\mathbf{Q}} z_t,$$

$$p^*_{i,t} = \mathbf{H}'_{(n-1)}\vec{\mathbf{x}}_t$$

Now, note that

$$\begin{aligned}
p_t &= \int_0^1 p_{i,t} di = \mathbf{H}'_{(n-1)} \int_0^1 \mathbb{E}_{i,t}[\vec{\mathbf{x}}_t] di \\
&= \mathbf{H}'_{(n-1)} \sum_{j=0}^{\infty} [(\mathbf{I} - \mathbf{K}_{(n)} \mathbf{Y}'_{(n)}) \mathbf{A}]^j \mathbf{K}_{(n)} \mathbf{Y}'_{(n)} \vec{\mathbf{x}}_{t-j} \\
&\approx \mathbf{H}'_{(n-1)} \underbrace{\left[\sum_{j=0}^{\infty} [(\mathbf{I} - \mathbf{K}_{(n)} \mathbf{Y}'_{(n)}) \mathbf{A}]^j \mathbf{K}_{(n)} \mathbf{Y}'_{(n)} \mathbf{M}'^j \right]}_{\equiv \mathbf{X}_{(n)}} \vec{\mathbf{x}}_t \\
&= \mathbf{H}'_{(n-1)} \mathbf{X}_{(n)} \vec{\mathbf{x}}_t = \mathbf{H}'_p \vec{\mathbf{x}}_t
\end{aligned}$$

Let $x_t = \mathbf{H}x'\vec{x}t$, $i_t = \mathbf{H}i'\vec{x}t$, and $\pi_t = \mathbf{H}\pi'\vec{x}t = \mathbf{H}'_p(\mathbf{I} - \mathbf{\Lambda M}')^{-1}(\mathbf{I} - \mathbf{M}')\vec{x}_t$. Then from the households Euler equation, we have:

$$\begin{aligned}
x_t &= \mathbb{E}_t^f \left[x_{t+1} - \frac{1}{\sigma} (i_t - \pi_{t+1}) \right] + \mathbb{E}_t^f [y_{t+1}^n] - y_t^n \\
\implies \mathbf{H}_i &= \sigma \left(\mathbf{M}' - \mathbf{I} \right) \mathbf{H}_x + \frac{\sigma(1+\psi)}{\sigma+\psi} \left(\mathbf{M}' - \mathbf{I} \right) \mathbf{H}_a + \mathbf{M}'\mathbf{H}_\pi
\end{aligned}$$

Also, the Talyor rule gives:

$$\begin{aligned}
& i_t = \rho_1 i_{t-1} + \rho_2 i_{t-2} + (1 - \rho_1 - \rho_2) \left(\phi_\pi \pi_t + \phi_x x_t + \phi_{\Delta y} \left(y_t - y_{t-1} \right) \right) + u_t \\
\implies & \left(\mathbf{I} - \rho_1 \mathbf{M} - \rho_2 \mathbf{M}^2 \right) \mathbf{H}_i = (1 - \rho_1 - \rho_2) \phi_\pi \mathbf{H}_\pi + (1 - \rho_1 - \rho_2) \phi_x \mathbf{H}_x \\
& + (1 - \rho_1 - \rho_2) \phi_{\Delta y} \left(\mathbf{I} - \mathbf{M} \right) \left(\mathbf{H}_x + \frac{1+\psi}{\sigma+\psi} \mathbf{H}_a \right) + \mathbf{H}_u
\end{aligned}$$

These give us $\mathbf{H}x$ and $\mathbf{H}i$ and we update new $\mathbf{H}_{(n)}$ using:

$$\mathbf{H}_{(n)} = \mathbf{H}_p + \alpha(\mathbf{I} - \mathbf{M\Lambda}')\mathbf{H}_x$$

We iterate until convergence of $\mathbf{H}_{(n)}$.

2.6.4 Replication Codes

Initialize:

```
using DRIPs;
using BenchmarkTools, LinearAlgebra, GLM, Statistics, Suppressor, Printf;
using PyPlot; rc("text", usetex="True") ;
              rc("font",family="serif",serif=:"Palatino") ;
using Plots, LaTeXStrings; pyplot() ;
```

Set parameters:

```
struct param
    β; σ; ψ; θ; α; # Deep parameters
    psi_π; psi_x; psi_dy; ρ; # Monetary policy parameters
    ρa; ρu; σa; σu; # Shock paramters
end
```

Assign model parameters:

```
σ       = 2.5       ; #Risk aversion
β       = 0.99      ; #Time discount
ψ       = 2.5       ; #Inverse of Frisch elasticity of labor supply
θ       = 10        ; #Elasticity of substitution across firms
α       = (σ+ψ)/(1+ψ*θ) ; #Strategic complementarity (1-α)
```

Assign monetary policy parameters: post-Volcker

```
psi_π     = 2.028     ; #Taylor rule response to inflation
psi_x     = 0.673/4   ; #Taylor rule response to output
psi_dy    = 3.122     ; #Taylor rule response to growth
ρ       = 0.9457    ; #interest rate smoothing
```

Assign monetary policy parameters: pre-Volcker

```
psi_π_pre = 1.589     ; #Taylor rule response to inflation
psi_x_pre = 1.167/4   ; #Taylor rule response to output
psi_dy_pre= 1.028     ; #Taylor rule response to output growth
ρ_pre   = 0.9181    ; #interest rate smoothing
```

Assign parameters governing shock processes

```
ρu      = 0.0       ; #persistence of MP shock (post-Volcker)
σu      = 0.279     ; #S.D. of MP shock (post-Volcker)
ρu_pre  = 0.0       ; #persistence of MP shock (pre-Volcker)
σu_pre  = 0.535     ; #S.D. of MP shock (pre-Volcker)
```

Calibrated parameters

```
ρa      = 0.85      ; #persistence of technology shock
σa      = 1.56      ; #S.D. of technology shock
ω       = 0.773     ; #marginal cost of information
```

Parameters for simulation/irfs

```
simT    = 50000     ;
nburn   = 500       ;
T       = 20        ;
```

Primitives of Drip

```
numshock= 2         ; #number of shocks
L       = 160       ; #length of trunction
M       = [zeros(1,L-1) 0; Matrix(I,L-1,L-1) zeros(L-1,1)];
M       = M^2       ;
J       = zeros(L,L); J[2,2] = 1 ;
Lambda  = zeros(L,L);
for i = 1:Int64(L/numshock); Lambda[i*numshock,i*numshock] = 1; end
A       = M + J     ;
eye     = Matrix(I,L,L);
```

We start with a function that solves the GE problem and returns the solution in a Drip structure:

```
function agg_drip(p::param,ω,A,Q,M,Lambda,
                  Ha,               #state space rep. of a
                  Hu;               #state space rep. of u
                  H0    = Hu+Ha,    #optional: initial guess for H0
                  Sigma = A*A'+Q*Q',#optional: initial guess for Σ_0
                  Omega = H0*H0',   #optional: initial guess for Ω
                  maxit = 10000,    #optional: max. iterations for GE code
                  maxit_in = 100,   #optional: max. iterations for solving DRIP
                  tol   = 1e-4,     #optional: tolerance for iterations
                  w     = 1)        #optional: update weight for RI
    err   = 1; iter  = 1; L = length(H0); eye = Matrix(I,L,L);
    temp0 = (eye-M)*inv(eye - M*Lambda') ;
    Htemp1 = (p.σ*(eye-p.ρ*M)*(M'-eye)-(1-p.ρ)*(p.psi_x*eye+p.psi_dy*(eye-M))) ;
    Htemp2 = (1-p.ρ)*p.psi_π*eye - (eye-p.ρ*M)*M' ;
    Htemp3 = (1+p.ψ)/(p.σ+p.ψ)*((1-p.ρ)*p.psi_dy*(eye-M)-p.σ*(eye-p.ρ*M)*(M'-eye));
    while (err > tol) & (iter < maxit)
        if iter == 1
            global ge  = Drip(ω,p.β,A,Q,H0;
                              Ω0=Omega, Σ0=Sigma, w=w, maxit=maxit_in);
        else
            global ge  = Drip(ω,p.β,A,Q,H0;
                              Ω0=ge.ss.Ω, Σ0=ge.ss.Σ_1, w=w, maxit=maxit_in);
        end
        XFUN(jj) = ((eye-ge.ss.K*ge.ss.Y')*ge.A)^jj*(ge.ss.K*ge.ss.Y')*(M')^jj
        X = DRIPs.infinitesum(XFUN; maxit=200, start = 0);
        global Hp = X'*H0 ;
        global Hπ = temp0*Hp ;
        Hπ[L-20:end,:] .= 0
        global Hx = (Htemp1)\(Htemp2*Hπ + Htemp3*Ha + Hu) ;
        global H1 = Hp + p.α*(eye - M*Lambda')*Hx ;
        err= 0.5*norm(H1-H0,2)/norm(H0)+0.5*err;
        H0 = H1;
        cap= DRIPs.capacity(ge, unit = "bit")
        if iter == maxit
            print("***GE loop hit maxit - no convergence\n")
        elseif mod(iter,50) == 0
            println("  Iteration $iter. Error: $err. Capacity: $cap.")
        end
        iter += 1;
    end
    return(ge,H1,Hx,Hp,Hπ)
end;
```

Model solution for post-Volcker Calibration:

```
Q       = zeros(L,2); Q[1,1]=σa; Q[2,2]=σu;
Ha_post = ρa.^(0:1:L/2-1) ;
Hu_post = ρu.^(0:1:L/2-1) ;
Ha_post = kron(Ha_post,[1,0])[:,:] ;
Hu_post = kron(Hu_post,[0,1])[:,:] ;
p       = param(β,σ,ψ,θ,α,psi_π,psi_x,psi_dy,ρ,ρa,ρu,σa,σu) ;
```

Get initial guess:

```
print("\nGet initial guess for post-Volcker solutions\n")
@time begin
@suppress agg_drip(p,ω,A,Q,M,Lambda,Ha_post,Hu_post;
w=0.95, maxit=300, maxit_in=5);
end;
```

Solve the model for the post-Volcker calibration:

```
print("\nSolve for the post-Volcker model:\n");
@time (ge_post,H1_post,Hx_post,Hp_post,Hπ_post) =
agg_drip(p,ω,A,Q,M,Lambda,Ha_post,Hu_post;
H0    = H1,
Sigma = ge.ss.Σ_1,
Omega = ge.ss.Ω,
w=0.95, maxit=5000, maxit_in=500) ;

Hy_post = Hx_post + (1+ψ)/(σ+ψ)*Ha_post      ;
Hi_post = M'*Hπ_post + σ*(M'-eye)*Hy_post    ;
Hr_post = Hi_post - (M')*Hπ_post             ;
```

Model solution for pre-Volcker Calibration:

```
Q_pre    = zeros(L,2); Q_pre[1,1]=σa; Q_pre[2,2]=σu_pre;
Ha_pre   = ρa.^(0:1:L/2-1)          ;
Hu_pre   = ρu_pre.^(0:1:L/2-1)      ;
Ha_pre   = kron(Ha_pre,[1,0])[:,:]  ;
Hu_pre   = kron(Hu_pre,[0,1])[:,:]  ;
p_pre    = param(β,σ,ψ,θ,α,psi_π_pre,psi_x_pre,psi_dy_pre,
ρ_pre,ρa,ρu_pre,σa,σu_pre) ;
```

Get initial guess:

```
print("\nGet initial guess for pre-Volcker solutions\n")
@time begin
@suppress agg_drip(p_pre,ω,A,Q_pre,M,Lambda,Ha_pre,Hu_pre;
                   H0    = H1_post,
                   Sigma = ge_post.ss.Σ_1,
                   Omega = ge_post.ss.Ω,
                   w=0.95, maxit=300, maxit_in=5) ;
end;
```

Solve the model for the pre-Volcker calibration:

```
print("\nSolve for the pre-Volcker model:\n");
@time begin (ge_pre,H1_pre,Hx_pre,Hp_pre,Hπ_pre) =
             agg_drip(p_pre,ω,A,Q_pre,M,Lambda,Ha_pre,Hu_pre;
                      H0    = H1,
                      Sigma = ge.ss.Σ_1,
                      Omega = ge.ss.Ω,
                      w=0.95, maxit=5000, maxit_in=500) ;
end

Hy_pre    = Hx_pre + (1+ψ)/(σ+ψ)*Ha_pre     ;
Hi_pre    = M'*Hπ_pre + σ*(M'-eye)*Hy_pre   ;
Hr_pre    = Hi_pre - (M')*Hπ_pre            ;
```

Model Simulation for post-Volcker:

```
print("\nSimulate the models:\n");

@time begin
    sim_post    = simulate(ge_post; T=simT, burn=nburn, seed=1) ;

    x_shock     = sim_post.x        ;
    xhat_avg    = sim_post.x_hat    ;

    sim_π       = (Hπ_post'*(eye-Lambda*M')*x_shock)' ;
    sim_y       = (Hy_post'*(eye-Lambda*M')*x_shock)' ;
    sim_x       = (Hx_post'*(eye-Lambda*M')*x_shock)' ;

    mat_sim     = [sim_π sim_y sim_x]   ;
    cor_sim     = cor(mat_sim)          ;

    stat_post   = vec([std(sim_π/100) std(sim_y/100) cor_sim[2,1]]) ;

    s = @sprintf("==> Post-Volcker: std(π)=%5.3f, std(y)=%5.3f, corr(π,y)=%5.3f",
    stat_post[1], stat_post[2], stat_post[3])  ;
    println(s) ;
end
```

```
Simulate the models:
==> Post-Volcker: std(π)=0.015, std(y)=0.018, corr(π,y)=0.209
```

Model Simulation for pre-Volcker:

```
@time begin
    sim_pre     = simulate(ge_pre; T=simT, burn=nburn, seed=1) ;

    x_shock_pre = sim_pre.x     ;
    xhat_avg_pre= sim_pre.x_hat ;

    sim_π_pre   = (Hπ_pre'*(eye-Lambda*M')*x_shock_pre)';
    sim_y_pre   = (Hy_pre'*(eye-Lambda*M')*x_shock_pre)';
    sim_x_pre   = (Hx_pre'*(eye-Lambda*M')*x_shock_pre)';

    mat_sim_pre = [sim_π_pre sim_y_pre sim_x_pre]   ;
    cor_sim_pre = cor(mat_sim_pre)                  ;

    stat_pre    = vec([std(sim_π_pre/100) std(sim_y_pre/100) cor_sim_pre[2,1]]);

    s = @sprintf("==> Pre-Volcker : std(π)=%5.3f, std(y)=%5.3f, corr(π,y)=%5.3f",
    stat_pre[1],stat_pre[2],stat_pre[3])  ;
    println(s) ;
end
```

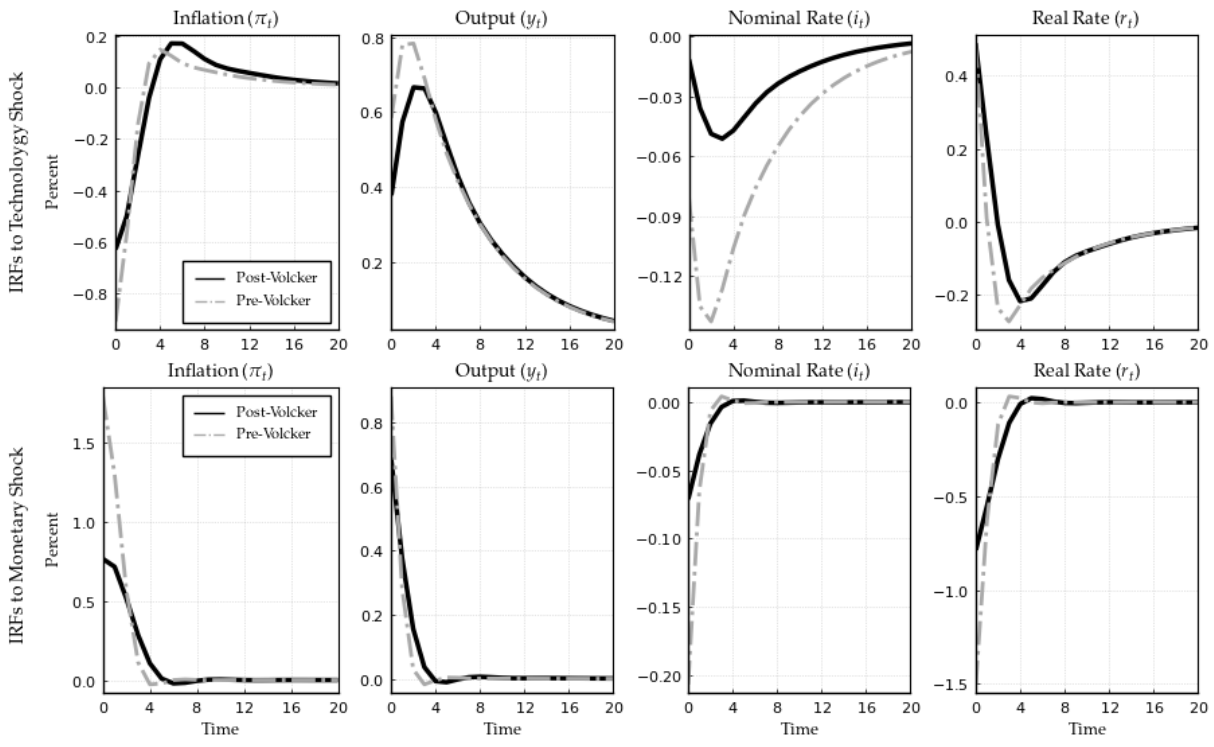
```
==> Pre-Volcker : std(π)=0.025, std(y)=0.020, corr(π,y)=0.245
```

IRFs:

```
pi      = reshape(Hπ_post,numshock,Int64(L/numshock))' ;
x       = reshape(Hx_post,numshock,Int64(L/numshock))' ;
y       = reshape(Hy_post,numshock,Int64(L/numshock))' ;
i       = reshape(Hi_post,numshock,Int64(L/numshock))' ;
r       = reshape(Hr_post,numshock,Int64(L/numshock))' ;
a       = reshape(Ha_post,numshock,Int64(L/numshock))' ;
u       = reshape(Hu_post,numshock,Int64(L/numshock))' ;

pi_pre  = reshape(Hπ_pre,numshock,Int64(L/numshock))' ;
x_pre   = reshape(Hx_pre,numshock,Int64(L/numshock))' ;
y_pre   = reshape(Hy_pre,numshock,Int64(L/numshock))' ;
i_pre   = reshape(Hi_pre,numshock,Int64(L/numshock))' ;
r_pre   = reshape(Hr_pre,numshock,Int64(L/numshock))' ;
a_pre   = reshape(Ha_pre,numshock,Int64(L/numshock))' ;
u_pre   = reshape(Hu_pre,numshock,Int64(L/numshock))' ;
```

```
title1 = Plots.plot(ylabel = "IRFs to Technoloygy Shock",
         guidefont = font(10), grid = false, showaxis = false,
         bottom_margin = 20Plots.px)
title2 = Plots.plot(ylabel = "IRFs to Monetary Shock",
         guidefont = font(10), grid = false, showaxis = false,
         bottom_margin = -50Plots.px, top_margin = 30Plots.px)
p1 = Plots.plot(0:T,[σa*pi[1:T+1,1],σa*pi_pre[1:T+1,1]],
     title  = L"Inflation ($\pi_t$)", ylabel = "Percent", guidefont = font(9),
     yticks = -1:0.2:0.3, label = ["Post-Volcker" "Pre-Volcker"],
     legend = :bottomright, legendfont= font(8),
     color  = [:black :darkgray], linestyle = [:solid :dashdot])
p2 = Plots.plot(0:T,[σa*y[1:T+1,1],σa*y_pre[1:T+1,1]],
     title  = L"Output ($y_t$)", legend = false,
     color  = [:black :darkgray], linestyle = [:solid :dashdot])
p3 = Plots.plot(0:T,[σa*i[1:T+1,1],σa*i_pre[1:T+1,1]],
     title  = L"Nominal Rate ($i_t$)", legend = false,
     yticks = -0.15:0.03:0.0, color = [:black :darkgray],
     linestyle = [:solid :dashdot])
p4 = Plots.plot(0:T,[σa*r[1:T+1,1],σa*r_pre[1:T+1,1]],
     title  = L"Real Rate ($r_t$)", legend = false,
     color  = [:black :darkgray], linestyle = [:solid :dashdot])
p5 = Plots.plot(0:T,[-σu*pi[1:T+1,2],-σu_pre*pi_pre[1:T+1,2]],
     title  = L"Inflation ($\pi_t$)", ylabel = "Percent",
     xlabel = "Time", guidefont = font(9), label = ["Post-Volcker" "Pre-Volcker"],
     legend = :topright, legendfont = font(8),
     color  = [:black :darkgray], linestyle = [:solid :dashdot])
p6 = Plots.plot(0:T,[-σu*y[1:T+1,2],-σu_pre*y_pre[1:T+1,2]],
     title    = L"Output ($y_t$)", xlabel = "Time",
     guidefont = font(9), legend = false,
     color     = [:black :darkgray], linestyle = [:solid :dashdot])
p7 = Plots.plot(0:T,[-σu*i[1:T+1,2],-σu_pre*i_pre[1:T+1,2]],
     title     = L"Nominal Rate ($i_t$)", xlabel = "Time",
     guidefont = font(9), legend = false,
     color     = [:black :darkgray], linestyle = [:solid :dashdot])
p8 = Plots.plot(0:T,[-σu*r[1:T+1,2],-σu_pre*r_pre[1:T+1,2]],
     title     = L"Real Rate ($r_t$)", xlabel = "Time",
     guidefont = font(9), legend = false,
     color     = [:black :darkgray], linestyle = [:solid :dashdot])
```

```
l = @layout [
            a{0.001w} Plots.grid(1,4)
            a{0.001w} Plots.grid(1,4)
            ]
Plots.plot(title1,p1,p2,p3,p4,title2,p5,p6,p7,p8,
      layout      = l, gridstyle = :dot,
      gridalpha   = 0.2, lw = [2.5 2],
      titlefont   = font(10), xticks = (0:4:T),
      xlim        = (0,T), tickfont= font(8),
      size        = (900,550), framestyle = :box)
```



References

Afrouzi, Hassan, "Strategic Inattention, Inflation Dynamics, and the Non-Neutrality of Money," CESifo Working Paper Series 8218, CESifo 2020.

_ **and Choongryul Yang**, "Dynamic Rational Inattention and the Phillips Curve," 2019. Available at SSRN: https://ssrn.com/abstract=3465793 or http://dx.doi.org/10.2139/ssrn.3465793.

Maćkowiak, Bartosz and Mirko Wiederholt, "Optimal Sticky Prices under Rational Inattention," *The American Economic Review*, 2009, *99* (3), 769–803.

Mackowiak, Bartosz and Mirko Wiederholt, "Rational Inattention and the Business Cycle Effects of Productivity and News Shocks," 2020.

Maćkowiak, Bartosz, Filip Matějka, and Mirko Wiederholt, "Dynamic Rational Inattention: Analytical Results," *Journal of Economic Theory*, 2018, *176*, 650 – 692.

Sims, Christopher A., "Rational Inattention and Monetary Economics," in Benjamin M. Friedman and Michael Woodford, eds., *Handbook of Monetary Economics*, Vol. 3, Elsevier, 2010, pp. 155–81.

Woodford, Michael, "Imperfect Common Knowledge and the Effects of Monetary Policy," in Philippe Aghion, Roman Frydman, Joseph Stiglitz, and Michael Woodford, eds., *Knowledge, Information, and Expectations in Modern Macroeconomics: In Honor of Edmund S. Phelps*, Princeton University Press, 2003, pp. 25–58.