Computational Macroeconomics

Summer 2023

Week 6

Willi Mutschler
willi@mutschler.eu

Version: 1.0.0
Latest version available on: GitHub

Contents

1. Case Study Eichenbaum, Rebelo and Trabandt (2022, JEDC): Epidemics in the New Keynesian model **1**

A. Helper functions for New Keynesian Epidemic model **3**
A.1. ert_model_sticky.mod 3
A.2. ert_model_go_calibrate_pi.m 6
A.3. ert_model_plot_agg_results.m 7
A.4. ert_model_plot_by_type_results.m 10

A. Solutions **12**

1. Case Study Eichenbaum, Rebelo and Trabandt (2022, JEDC): Epidemics in the New Keynesian model

1. Read the paper by Eichenbaum, Rebelo, and Trabandt (2022) and focus particularly on Section 3 (The model economy), Section 4 (The impact of an epidemic), and the Appendix A (Equilibrium equations).

2. Consider the Dynare mod file given in Appendix A.1, which is a stripped down version of the sticky-price New Keynesian SIR model as outlined in Appendix A of the paper. Compute the steady-state using an `initval` block.

3. Note that the mod file contains a `predetermined_variables` command. Read about this in the manual of Dynare (Adjemian et al. 2022) and explain why this is relevant in the current model.

4. Note that the Taylor rule, in equation 31), exhibits differences between the mod file and the paper. Specifically, in the paper, the output gap is defined as the discrepancy between output and the corresponding output level under flexible prices ($\xi = 0$), whereas in the mod file, it is defined as deviations from the steady-state. Generate a new mod file that incorporates equations for both the sticky-price and flexible-price economies, with the two models connected through the Taylor rule in the sticky model. Additionally, you need to recalculate the pre-infectious steady-state for the combined economies. *Hint: Don't forget to set the macro directive by including:* `#define FLEX_PRICE_OUTPUT_GAP = 1`

5. Read section `4.7. Initial and terminal conditions` in the manual of Dynare (Adjemian et al. 2022). Add a `histval` block that implements the impact of an epidemic in the model, i.e. in period $t = 0$ change infected to $i_0 = \varepsilon$ and susceptibles to $s_0 = 1 - \varepsilon$, while all other variables are set to their pre-infection steady-state. Run `perfect_foresight_setup` and have a look how Dynare initializes `M_._endo_histval`, `oo_._endo_simul` and `oo._exo_simul` for the Newton algorithm.

6. Notice that the `model` block has a `differentiate_forward_vars` option. Look into the manual of Dynare (Adjemian et al. 2022) what this option does. Explain why it is necessary in this model when we want to simulate the impact of an epidemic?

7. What is *homotopy* in the context of perfect foresight simulations? Illustrate this by setting

$$\pi_1 = 2.568436063602094 \times 10^{-7}$$
$$\pi_2 = 1.593556989906377 \times 10^{-4}$$
$$\pi_3 = 0.499739472034583$$

   and running a perfect foresight solver using the above `histval` block.

8. Simulate the economy when the epidemic is a shock to consumption demand only, i.e. replicate Figures 5 and 6. To this end, calibrate the probabilities according to:

```
% calibation targets for shares of pi-terms in T-function in SIR model
pi3_shr_target = 2/3;                  % share of T_0 jump due general infections
pi1_shr_target = (1-pi3_shr_target);   % share of T_0 jump due to consumption-based infections
pi2_shr_target = 0;                    % share of T_0 jump due to work-based infections
[pi1_final,pi2_final,pi3_final] = ert_model_go_calibrate_pi(250,varepsilon,pir,pid,pi1_shr_target,
    pi2_shr_target,RplusD_target,c_ss,n_ss);
```

9. Simulate the economy when the epidemic is a shock to labor supply only, i.e. replicate Figures 7 and 8. To this end, calibrate the probabilities according to:

```
pi3_shr_target = 2/3;                  % share of T_0 jump due general infections
pi1_shr_target = 0;                    % share of T_0 jump due to consumption-based infections
pi2_shr_target = (1-pi3_shr_target);   % share of T_0 jump due to work-based infections
[pi1_final,pi2_final,pi3_final] = ert_model_go_calibrate_pi(250,varepsilon,pir,pid,pi1_shr_target,
    pi2_shr_target,RplusD_target,c_ss,n_ss);
```

10. Simulate the economy when the epidemic is a shock to both consumption demand and labor supply, i.e. replicate Figures 9 and 10. To this end, calibrate the probabilities according to:

```
pi3_shr_target = 2/3;                  % share of T_0 jump due general infections
pi1_shr_target = (1-pi3_shr_target)/2; % share of T_0 jump due to consumption-based infections
pi2_shr_target = (1-pi3_shr_target)/2; % share of T_0 jump due to work-based infections
[pi1_final,pi2_final,pi3_final] = ert_model_go_calibrate_pi(250,varepsilon,pir,pid,pi1_shr_target,
    pi2_shr_target,RplusD_target,c_ss,n_ss);
```

Hints

- Appendix A.2 contains a helper function to calibrate the probabilities π_1, π_2 and π_3.
- Appendix A.3 contains a helper function to plot figures 5, 7, and 9.
- Appendix A.4 contains a helper function to plot figures 6, 8, and 10.
- It might help to know that you can have multiple `model` and `var` blocks in the same mod file.
- Dynare's `#include "SOMEMODILE"` directive might be useful.
- `set_param_value('pi3',pi3_final)` is a useful function to update parameters when you loop over parameters.
- If your solution does not converge, try homotopy for π_3, e.g. start with $\pi_3/3$.
- Set the `periods` option to 500 and the `maxit` option to 100.

References

Adjemian, Stéphane et al. (Jan. 2022). *Dynare: Reference Manual Version 5*. Tech. rep. 72. CEPREMAP. URL: `https://ideas.repec.org/p/cpm/dynare/072.html` (visited on 01/21/2021).

Eichenbaum, Martin S., Sergio Rebelo, and Mathias Trabandt (July 2022). "Epidemics in the New Keynesian Model". In: *Journal of Economic Dynamics and Control* 140, p. 104334. DOI: `10.1016/j.jedc.2022.104334`.

A. Helper functions for New Keynesian Epidemic model

A.1. ert_model_sticky.mod

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_sticky.mod

```
% ========================================================================
% Stripped down sticky-price version of New Keynesian Model of Eichenbaum,
% Rebelo, and Trabandt (2022, JEDC): "Epidemics in the New Keynesian model"
% based on the original Dynare replication files kindly provided by the
% authors (downloaded from Mathias Trabandt's homepage)
% ------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

%-------------------------------------------------------------------------%
% DECLARE ENDOGENOUS VARIABLES FOR THE STICKY-PRICE ECONOMY
%-------------------------------------------------------------------------%
var
y           ${y}$                   (long_name='output')
k           ${k}$                   (long_name='aggregate capital')
n           ${n}$                   (long_name='aggregate labor')
w           ${w}$                   (long_name='real wage')
rk          ${r^k}$                 (long_name='real rental rate of capital')
x           ${x}$                   (long_name='investment')
c           ${c}$                   (long_name='aggregate consumption')
s           ${s}$                   (long_name='susceptible')
i           ${i}$                   (long_name='infected')
r           ${r}$                   (long_name='recovered')
ns          ${n^s}$                 (long_name='labor supply of susceptibles')
ni          ${n^i}$                 (long_name='labor supply of infected')
nr          ${n^r}$                 (long_name='labor supply of recovered')
cs          ${c^s}$                 (long_name='consumption of susceptibles')
ci          ${c^i}$                 (long_name='consumption of infected')
cr          ${c^r}$                 (long_name='consumption of recovered')
tau         ${\tau}$                (long_name='newly infected')
lambtilde   ${\tilde{\lambda}^b}$   (long_name='Lagrange multiplier on household budget')
lamtau      ${\lambda^\tau}$        (long_name='Lagrange multiplier on transmission function')
lami        ${\lambda^i}$           (long_name='Lagrange multiplier on law of motion of infected')
lams        ${\lambda^s}$           (long_name='Lagrange multiplier on law of motion of susceptible')
lamr        ${\lambda^r}$           (long_name='Lagrange multiplier on law of motion of recovered')
dd          ${d}$                   (long_name='deceased')
pop         ${pop}$                 (long_name='population')
pbreve      ${\breve{p}}$           (long_name='inverse price dispersion')
mc          ${mc}$                  (long_name='real marginal costs')
rr          ${rr}$                  (long_name='real interest rate')
Rb          ${R^b}$                 (long_name='nominal interest rate')
pie         ${\pi}$                 (long_name='inflation')
Kf          ${K^f}$                 (long_name='auxiliary variable 1 in recursive price setting')
F           ${F}$                   (long_name='auxiliary variable 2 in recursive price setting')
@#ifdef FLEX_PRICE_OUTPUT_GAP
y_flex      ${y^{flex}}$            (long_name='output (flex-price)')
@#endif
;

%-------------------------------------------------------------------------%
% DECLARE MODEL PARAMETERS
%-------------------------------------------------------------------------%
parameters
xi          ${\xi}$         (long_name='Calvo probability sticky-price economy')
rpi         ${r_\pi}$       (long_name='Taylor rule coefficient inflation')
rx          ${r_x}$         (long_name='Taylor rule coefficient output gap')
gam         ${\gamma}$      (long_name='elasticity in Dixit-Stiglitz aggregator (steady-state gross price markup)')
pi1         ${\pi_1}$       (long_name='probability of becoming infected as a result of time spent in consumption
    activities')
pi2         ${\pi_2}$       (long_name='probability of becoming infected as a result of time spent in working
    interactions')
pi3         ${\pi_3}$       (long_name='probability of becoming infected as a result of random meetings')
pir         ${\pi_r}$       (long_name='probability of recovering from infection')
pid         ${\pi_d}$       (long_name='probability of dying from infection')
betta       ${\beta}$       (long_name='discount factor')
A           ${A}$           (long_name='common productivity level')
theta       ${\theta}$      (long_name='weight of labor in utility function')
```

```
alfa        ${\alpha}$       (long_name='labor share in production')
delta       ${\delta}$       (long_name='capital depreciation rate')
g_ss        ${G}$            (long_name='government spending')
inc_target  ${\bar{Y}}$      (long_name='calibration target for income in steady-state')
n_target    ${\bar{N}}$      (long_name='calibration target for labor in steady-state')
eta         ${\eta}$         (long_name='calibration target for government consumption to GDP ratio')
varepsilon  ${\varepsilon}$  (long_name='initial seed of infection (I_0)')
;

%-----------------------------------------------------------------------------%
% STICKY-PRICE MODEL EQUATIONS (ACCORDING TO APPENDIX A)
%-----------------------------------------------------------------------------%
predetermined_variables k i s r dd pop;
model(differentiate_forward_vars);
[name='1) aggregate supply']
y = pbreve * A * k^(1-alfa) * n^alfa;
[name='2) marginal costs']
mc = w^alfa * rk^(1-alfa) / ( A * alfa^alfa * (1-alfa)^(1-alfa) );
[name='3) optimal factor input']
w = mc * alfa * A * n^(alfa-1) * k^(1-alfa);
[name='4) capital accumulation']
k(+1) = (1-delta) * k + x;
[name='5) aggregate demand']
y = c + x + g_ss;
[name='6) aggregate hours']
n = (s * ns) + (i * ni) + (r * nr);
[name='7) aggregate consumption']
c = (s * cs) + (i * ci) + (r * cr);
[name='8) transmission function for new infections']
tau = pi1 * (s * cs) * (i * ci) + pi2 * (s * ns) * (i * ni) + pi3 * (s * i);
[name='9) law of motion susceptible population']
s(+1) = s - tau;
[name='10) law of motion infected population']
i(+1) = i + tau - (pir + pid) * i;
[name='11) law of motion recovered population']
r(+1) = r + pir * i;
[name='12) law of motion deceased population']
dd(+1) = dd + pid * i;
[name='13) total population']
pop(+1) = pop - pid * i;
[name='14) first order condition wrt consumption susceptibles']
1 / cs = lambtilde - lamtau * pi1 * (i * ci);
[name='15) first order condition wrt consumption infected']
1 / ci = lambtilde;
[name='16) first order condition wrt consumption recovered']
1 / cr = lambtilde;
[name='17) first order condition wrt hours susceptibles']
theta * ns = lambtilde * w + lamtau * pi2 * (i * ni);
[name='18) first order condition wrt hours infected']
theta * ni = lambtilde * w;
[name='19) first order condition wrt hours recovered']
theta * nr = lambtilde * w;
[name='20) first order condition wrt capital']
lambtilde = betta * ( rk(+1) + 1 - delta ) * lambtilde(+1);
[name='21) first order condition wrt newly infected']
lami = lamtau + lams;
[name='22) first order condition wrt susceptibles']
0 = log(cs(+1)) - theta/2 * ns(+1)^2
  + lamtau(+1) * ( pi1 * cs(+1) * i(+1) * ci(+1)
                 + pi2 * ns(+1) * i(+1) * ni(+1)
                 + pi3 * i(+1)
                 )
  + lambtilde(+1) * ( w(+1) * ns(+1) - cs(+1) )
  - lams / betta + lams(+1);
[name='23) first order condition wrt infected']
0 = log(ci(+1)) - theta/2 * ni(+1)^2
  + lambtilde(+1) * ( w(+1) * ni(+1) - ci(+1) )
  - lami/betta + lami(+1) * (1 - pir - pid) + lamr(+1)*pir;
[name='24) first order condition wrt recovered']
0 = log(cr(+1)) - theta/2 * nr(+1)^2
  + lambtilde(+1) * ( w(+1) * nr(+1) -  cr(+1) )
  - lamr/betta + lamr(+1);
[name='25) first order condition wrt bonds']
lambtilde = betta * rr * lambtilde(+1);
[name='26) real interest rate']
```

```
rr = Rb / pie(+1);
[name='27) recursion nonlinear price setting 1']
Kf = gam * mc * lambtilde * y + betta * xi * pie(+1)^(gam/(gam-1)) * Kf(+1);
[name='28) recursion nonlinear price setting 2']
F = lambtilde * y + betta * xi * pie(+1)^(1/(gam-1)) *F(+1);
[name='29) nonlinear price setting']
Kf = F * ( ( 1 - xi * pie^(1/(gam-1)) ) / (1 - xi) )^(-(gam-1));
[name='30) inverse price dispersion']
pbreve^(-1) = (1-xi) * ( ( 1 - xi * pie^(1/(gam-1)) ) / (1 - xi) )^gam
            + xi * pie^(gam/(gam-1)) / pbreve(-1);
[name='31) Taylor rule']
Rb - STEADY_STATE(Rb) = rpi * log( pie / STEADY_STATE(pie) )
@#ifdef FLEX_PRICE_OUTPUT_GAP
                      + rx * log( y / y_flex )
@#else
                      + rx * log( y / steady_state(y) )
@#endif
;
end;

%-------------------------------------------------------------------------------%
% COMMON WEEKLY CALIBRATION (SECTION 3.1, TABLE 8)
%-------------------------------------------------------------------------------%
pid        = 7*0.002/14;  % probability of dying
pir        = 7/14 - pid;  % probability of recovering
delta      = 0.06/52;     % capital depreciation rate
alfa       = 2/3;         % labor share
gam        = 1.35;        % gross price markup
xi         = 0.98;        % Calvo price stickiness
rpi        = 1.5;         % Taylor rule coefficient inflation
rx         = 0.5/52;      % Taylor rule coefficient output gap
eta        = 0.19;        % government consumption share of output
n_target   = 28;          % hours worked target
inc_target = 58000/52;    % income target
betta      = 0.98^(1/52); % depreciation rate
varepsilon = 0.001;       % initial seed of infection (varepsilon)
RplusD_target = 0.60;     % total share of people infected and then either
                          % recovered or dead after epidemic

%-------------------------------------------------------------------------------%
% PRE-INFECTION STEADY-STATE
%-------------------------------------------------------------------------------%
pi1 = 0; pi2 = 0; pi3 = 0; % initialize to compute steady-state,
                           % these will change depending on epidemic type
pop_ss = 1; % normalize
s_ss = 1;   % no infections
i_ss = 0;   % no infections
r_ss = 0;   % no infections
dd_ss = 0;  % no infections
y_ss = inc_target; % calibration target
n_ss = n_target;   % calibration target
pie_ss = 1; % calibration target
pbreve_ss = 1; % law of motion price dispersion
mc_ss = 1/gam; % recursive price setting, combined with no inflation steady-state
w_ss  = mc_ss*alfa*y_ss/n_ss; % optimal labor demand
rk_ss = 1/betta-1+delta; % first-order condition wrt capital
kn_ss = (1-alfa)*w_ss/alfa/rk_ss; % optimal factor input (as a ratio to labor)
yk_ss = (y_ss/n_ss)/kn_ss; % definition output-to-capital ratio
A = (y_ss/n_ss)^alfa*yk_ss^(1-alfa); % production function
k_ss = (y_ss/A/n_ss^alfa)^(1/(1-alfa)); % production function
x_ss = delta*k_ss; % capital accumulation
g_ss = eta*y_ss; % target
c_ss = y_ss - x_ss - g_ss; % aggregate demand
ns_ss = n_ss; % aggregate labor with no infections
cs_ss = c_ss; % aggregate consumption with no infections
tau_ss = 0; % transmission function with no infections
s_ss = 1; % law of motion susceptibles with no infections
i_ss = 0; % law of motion infected with no infections
r_ss = 0; % law of motion recovered with no infections
lambtilde_ss = 1/cs_ss; % first-order condition wrt susceptibles
ci_ss = cs_ss; % first-order condition wrt infected
cr_ss = cs_ss; % first-order condition wrt recovered
theta = lambtilde_ss*w_ss/ns_ss; % first order condition wrt hours susceptibles
ni_ss = lambtilde_ss*w_ss/theta; % first order condition wrt hours infected
nr_ss = ns_ss; % first order condition wrt hours recovered
```

```
% value of life
lams_ss = (log(cs_ss)-theta/2*ns_ss^2 + lambtilde_ss*(w_ss*ns_ss-cs_ss) ) / ( 1/betta-1 );
lamr_ss = (log(cr_ss)-theta/2*nr_ss^2 + lambtilde_ss*(w_ss*nr_ss-cr_ss) ) / ( 1/betta-1 );
lami_ss = (log(ci_ss)-theta/2*ni_ss^2 + lambtilde_ss*(w_ss*ni_ss-ci_ss) + pir*lamr_ss) / ( 1/betta-1+pir+pid );
lamtau_ss = lami_ss-lams_ss; % first order condition wrt newly infected
Rb_ss = pie_ss/betta; % first order condition wrt bonds
rr_ss = Rb_ss/pie_ss; % real interest rate
% recursion nonlinear price setting (sticky-price)
Kf_ss  = 1/(1-betta*xi)*gam*mc_ss*lambtilde_ss*y_ss;
F_ss   = 1/(1-betta*xi)*lambtilde_ss*y_ss;

%-------------------------------------------------------------------------%
% display useful calibration targets, ratios and steady-state
%-------------------------------------------------------------------------%
fprintf('CALIBRATION TARGETS, RATIOS, STEADY-STATE\n');
fprintf('  - value of life = %.2f\n', 1/(1-betta)*(log(c_ss)-theta/2*n_ss^2)*c_ss);
fprintf('  - steady-state real wage = %.2f\n', w_ss);
fprintf('  - value of utility weight of labor = %.2f\n', theta);
fprintf('  - value of common productivtiy level = %.3f\n', A);
fprintf('  - annualized capital to output ratio = %.2f%%\n', k_ss/(52*y_ss));
fprintf('  - share of investment as a fraction of y = %.2f%%\n',100*x_ss/y_ss);
fprintf('  - share of consumption as a fraction of y = %.2f%%\n',100*c_ss/y_ss);
fprintf('  - share of gov. spending as a fraction of y = %.2f%%\n',100*g_ss/y_ss);
fprintf('  - prices change on average %.2f times per year\n', (1/(1-xi))/52)
```

A.2. ert_model_go_calibrate_pi.m

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_go_calibrate_pi.m

```
function [pi1,pi2,pi3] = ert_model_go_calibrate_pi(HH,varepsilon,pir,pid,pi1_shr_target,pi2_shr_target,RplusD_target,
    c_ss,n_ss)
% function [pi1,pi2,pi3] = ert_model_go_calibrate_pi(HH,varepsilon,pir,pid,pi1_shr_target,pi2_shr_target,RplusD_target
    ,c_ss,n_ss)
%-------------------------------------------------------------------------
% calibrates probabilities to get infected in the SIR model in
% Eichenbaum, Rebelo, Trabandt (2022, JEDC): "Epidemics in the New Keynesian model"
% based on go_calibrate_pi.m and calibrate_pi.m from the original Dynare replication files
% kindly provided by the authors (downloaded from Mathias Trabandt's homepage)
%-------------------------------------------------------------------------
% INPUTS
% - HH              [integer]  horizon of simulation
% - varepsilon      [double]   initial seed of infection
% - pir             [double]   probability of recovering from infection
% - pid             [double]   probability of dying from infection
% - pi1_shr_target  [double]   share of T_0 jump due to consumption-based infections
% - pi2_shr_target  [double]   share of T_0 jump due to work-based infections
% - RplusD_target   [double]   total share of people infected and then either recovered or dead after epidemic
% - c_ss            [double]   pre-infection steady-state of consumption
% - n_ss            [double]   pre-infection steady-state of hours worked
%-------------------------------------------------------------------------
% OUTPUTS
% - pi1             [double]   probability of becoming infected as a result of time spent in consumption activities
% - pi2             [double]   probability of becoming infected as a result of time spent in working interactions
% - pi3             [double]   probability of becoming infected as a result of random meetings
%-------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% =========================================================================

% pi1, pi2, and pi3 have different scales, so we re-scale pi1 and pi2
% such that they have the same scale as pi3 (e.g. pi3=0.4997)
% we then optimize over [pi1*scale1; pi2*scale2; pi3]
scale1 = 1e6; % scale of pi1 for numerical solver as we expect very tiny pi1,
              % e.g. pi1=2.5648*1e-7 and pi1*scale1=0.2565
scale2 = 1e3; % scale of pi2 for numerical solver as we expect very small pi2,
              % e.g. pi2=1.5936*1e-4 and then pi2*scale2=0.1594

opts_fsolve = optimoptions('fsolve','Display','iter','TolFun',1e-7,'MaxFunctionEvaluations',1e5,'MaxIterations',4e3);

[sol,~,exitflag] = fsolve(@calibrate_pi, [0.2;0.2;0.2], opts_fsolve,...
                          HH,varepsilon,pir,pid,pi1_shr_target,pi2_shr_target,RplusD_target,c_ss,n_ss,scale1,scale2);
if exitflag~=1
    error('fsolve could not calibrate the SIR model');
```

```
else
    [err,pi1,pi2,pi3,RnotSIR] = calibrate_pi(sol,HH,varepsilon,pir,pid,pi1_shr_target,pi2_shr_target,RplusD_target,
        c_ss,n_ss,scale1,scale2);

    disp(['Max. abs. error in calibration targets:',num2str(max(abs(err)))]);
    disp([' ']);
    pi1=sol(1)/scale1
    pi2=sol(2)/scale2
    pi3=sol(3)
    RnotSIR
end

% helper function
function [err,pi1,pi2,pi3,RnotSIR,I,S,D,R,T] = calibrate_pi(pi_guess,HH,varepsilon,pir,pid,pi1_shr_target,
     pi2_shr_target,RplusD_target,C_ss,N_ss,scale1,scale2)
%back out initial guesses
pi1=pi_guess(1)/scale1; pi2=pi_guess(2)/scale2; pi3=pi_guess(3);
%pre-allocate
I=NaN*ones(HH+1,1); S=NaN*ones(HH+1,1); D=NaN*ones(HH+1,1); R=NaN*ones(HH+1,1); T=NaN*ones(HH,1);
%initial conditions
I(1)=varepsilon; S(1)=1-I(1); D(1)=0; R(1)=0;
%iterate on SIR model equations
for j=1:1:HH
    T(j,1)=pi1*S(j)*C_ss^2*I(j)+pi2*S(j)*N_ss^2*I(j)+pi3*S(j)*I(j);
    S(j+1,1)=S(j)-T(j);
    I(j+1,1)=I(j)+T(j)-(pir+pid)*I(j);
    R(j+1,1)=R(j)+pir*I(j);
    D(j+1,1)=D(j)+pid*I(j);
end
%calculate equation residuals for target equations (16) and (17) in paper
err(1) = pi1_shr_target-(pi1*C_ss^2)/(pi1*C_ss^2+pi2*N_ss^2+pi3);
err(2) = pi2_shr_target-(pi2*N_ss^2)/(pi1*C_ss^2+pi2*N_ss^2+pi3);
err(3) = RplusD_target-(R(end)+D(end));
RnotSIR=T(1)/I(1)/(pir+pid);
end % calibrate_pi end

end % Eichenbaum_Rebelo_Trabandt_2022_go_calibrate_pi end
```

A.3. ert_model_plot_agg_results.m

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_plot_agg_results.m

```
function ert_model_plot_agg_results(epidemic_type,M_,oo_,flex_price)
% function ert_model_plot_agg_results(epidemic_type,M_,oo_,flex_price)
% ------------------------------------------------------------------------
% create plots of aggregate variables from the New Keynesian Model of
% Eichenbaum, Rebelo, Trabandt (2022, JEDC): "Epidemics in the New Keynesian model"
% based on plot_agg_results.m from the original Dynare replication files
% kindly provided by the authors (downloaded from Mathias Trabandt's homepage)
% ------------------------------------------------------------------------
% INPUTS
% - epidemic_type  [string]     type of epidemic, possible values:
%                               "demand", "supply", "both"
% - M_             [structure]  Dynare's model structure
% - oo_            [structure]  Dynare's results structure
% - flex_price     [boolean]    indicator whether to also plot flex-price model
% ------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

% common settings for plots
fsize = 12;
horz  = 100;
time  = 0:1:horz-1;

% title for figure
if epidemic_type == "demand"
    strtitle = "Epidemic as a shock to consumption demand.";
elseif epidemic_type == "supply"
    strtitle = "Epidemic as a shock to labor supply.";
elseif epidemic_type == "both"
    strtitle = "Epidemic as a shock to both consumption demand and labor supply.";
```

```
end
figure('name',strtitle,'units','normalized','outerposition',[0 0 0.75 0.75]);

% Nominal Interest Rate
Rb_ss = oo_.steady_state(M_.endo_names=="Rb");
Rb = oo_.endo_simul(M_.endo_names=="Rb",:);
subplot(3,3,1);
hold on;
plot(time(1:end-1),0*100*((Rb(2:horz)).^52-1)+100*(Rb_ss.^52-1),'m:','LineWidth',1.5);
plot(time(1:end-1),100*((Rb(2:horz)).^52-1),'b-','LineWidth',2);
if flex_price
    Rb_flex = oo_.endo_simul(M_.endo_names=="Rb_flex",:);
    plot(time(1:end-1),100*((Rb_flex(2:horz)).^52-1),'r--','LineWidth',2);
end
hold off;
box off;
title('Nominal Interest Rate','FontSize',fsize);
xlabel('Weeks','FontSize',fsize); xticks(0:20:80);
ylabel('in %');
set(gca,'FontSize',fsize);
axis tight;

% Inflation
pie_ss = oo_.steady_state(M_.endo_names=="pie");
pie = oo_.endo_simul(M_.endo_names=="pie",:);
subplot(3,3,2);
hold on;
plot(time(1:end-1),0*100*((pie(2:horz)).^52-1)+100*(pie_ss.^52-1),'m:','LineWidth',1.5);
plot(time(1:end-1),100*((pie(2:horz)).^52-1),'b-','LineWidth',2);
if flex_price
    pie_flex = oo_.endo_simul(M_.endo_names=="pie_flex",:);
    plot(time(1:end-1),100*((pie_flex(2:horz)).^52-1),'r--','LineWidth',2);
end
hold off;
box off;
title('Inflation','FontSize',fsize);
xlabel('Weeks','FontSize',fsize); xticks(0:20:80);
ylabel('in %')
set(gca,'FontSize',fsize);
axis tight;

% GDP
y_ss = oo_.steady_state(M_.endo_names=="y");
y = oo_.endo_simul(M_.endo_names=="y",:);
subplot(3,3,3);
hold on;
plot(time(1:end-1),0*100*(y(2:horz)-y_ss)/y_ss,'m:','LineWidth',1.5);
plot(time(1:end-1),100*(y(2:horz)-y_ss)/y_ss,'b-','LineWidth',2);
if flex_price
    y_flex = oo_.endo_simul(M_.endo_names=="y_flex",:);
    plot(time(1:end-1),100*(y_flex(2:horz)-y_ss)/y_ss,'r--','LineWidth',2);
end
hold off;
box off;
title('GDP','FontSize',fsize);
xlabel('Weeks','FontSize',fsize); xticks(0:20:80);
ylabel('in % dev from init ss')
set(gca,'FontSize',fsize);
axis tight;

% Consumption
c_ss = oo_.steady_state(M_.endo_names=="c");
c = oo_.endo_simul(M_.endo_names=="c",:);
subplot(3,3,4);
hold on;
plot(time(1:end-1),0*100*(c(2:horz)-c_ss)/c_ss,'m:','LineWidth',1.5);
plot(time(1:end-1),100*(c(2:horz)-c_ss)/c_ss,'b-','LineWidth',2);
if flex_price
    c_flex = oo_.endo_simul(M_.endo_names=="c_flex",:);
    plot(time(1:end-1),100*(c_flex(2:horz)-c_ss)/c_ss,'r--','LineWidth',2);
end
hold off;
box off;
title('Consumption','FontSize',fsize);
xlabel('Weeks','FontSize',fsize); xticks(0:20:80);
```

```
ylabel('in % dev from init ss')
set(gca,'FontSize',fsize);
axis tight;

% Hours
n_ss = oo_.steady_state(M_.endo_names=="n");
n = oo_.endo_simul(M_.endo_names=="n",:);
subplot(3,3,5);
hold on;
plot(time(1:end-1),0*100*(n(2:horz)-n_ss)/n_ss,'m:','LineWidth',1.5);
plot(time(1:end-1),100*(n(2:horz)-n_ss)/n_ss,'b-','LineWidth',2);
if flex_price
    n_flex = oo_.endo_simul(M_.endo_names=="n_flex",:);
    plot(time(1:end-1),100*(n_flex(2:horz)-n_ss)/n_ss,'r--','LineWidth',2);
end
hold off;
box off;
title('Hours','FontSize',fsize);
xlabel('Weeks','FontSize',fsize); xticks(0:20:80);
ylabel('in % dev from init ss')
set(gca,'FontSize',fsize);
axis tight;

% Investment
x_ss = oo_.steady_state(M_.endo_names=="x");
x = oo_.endo_simul(M_.endo_names=="x",:);
subplot(3,3,6);
hold on;
plot(time(1:end-1),0*100*(x(2:horz)-x_ss)/x_ss,'m:','LineWidth',1.5);
plot(time(1:end-1),100*(x(2:horz)-x_ss)/x_ss,'b-','LineWidth',2);
if flex_price
    x_flex = oo_.endo_simul(M_.endo_names=="x_flex",:);
    plot(time(1:end-1),100*(x_flex(2:horz)-x_ss)/x_ss,'r--','LineWidth',2);
end
hold off;
box off;
title('Investment','FontSize',fsize);
xlabel('Weeks','FontSize',fsize); xticks(0:20:80);
ylabel('in % dev from init ss')
set(gca,'FontSize',fsize);
axis tight;


% Real Interest Rate
rr_ss = oo_.steady_state(M_.endo_names=="rr");
rr = oo_.endo_simul(M_.endo_names=="rr",:);
subplot(3,3,7);
hold on;
plot(time(1:end-1),0*100*((rr(2:horz)).^52-1)+100*(rr_ss.^52-1),'m:','LineWidth',1.5);
plot(time(1:end-1),100*((rr(2:horz)).^52-1),'b-','LineWidth',2);
if flex_price
    rr_flex = oo_.endo_simul(M_.endo_names=="rr_flex",:);
    plot(time(1:end-1),100*((rr_flex(2:horz)).^52-1),'r--','LineWidth',2);
end
hold off;
box off;
title('Real Interest Rate','FontSize',fsize);
xlabel('Weeks','FontSize',fsize); xticks(0:20:80);
ylabel('in %');
set(gca,'FontSize',fsize);
axis tight;

% Infected
i = oo_.endo_simul(M_.endo_names=="i",:);
subplot(3,3,8);
hold on;
plot(time,100*i(1:horz),'b-','LineWidth',2);
if flex_price
    i_flex = oo_.endo_simul(M_.endo_names=="i_flex",:);
    plot(time,100*i_flex(1:horz),'r--','LineWidth',2);
end
hold off;
box off;
title('Infected','FontSize',fsize);
xlabel('Weeks','FontSize',fsize); xticks(0:20:80);
```

```
ylabel('in % dev from init pop')
set(gca,'FontSize',fsize);
axis tight;

% Deaths
dd = oo_.endo_simul(M_.endo_names=="dd",:);
subplot(3,3,9);
hold on;
plot(time,100*dd(1:horz),'b-','LineWidth',2);
if flex_price
    dd_flex = oo_.endo_simul(M_.endo_names=="dd_flex",:);
    plot(time,100*dd_flex(1:horz),'r--','LineWidth',2);
end
hold off;
box off;
title('Deaths','FontSize',fsize);
xlabel('Weeks','FontSize',fsize); xticks(0:20:80);
ylabel('in % dev from init pop')
set(gca,'FontSize',fsize);
axis tight;

if flex_price
    legend('New Keynesian Model (Sticky Prices)','Model with Flexible Prices','Location','best');
    legend boxoff
end

if epidemic_type == "demand"
    suptitle("Figure 5: " + strtitle);
elseif epidemic_type == "supply"
    suptitle("Figure 7: " + strtitle);
elseif epidemic_type == "both"
    suptitle("Figure 9: " + strtitle);
end
```

A.4. ert_model_plot_by_type_results.m

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_plot_by_type_results.m

```
function ert_model_plot_by_type_results(epidemic_type,M_,oo_)
% function ert_model_plot_by_type_results(epidemic_type,M_,oo_)
%-------------------------------------------------------------------------
% create plots of consumption and hours by population type from the New Keynesian Model of
% Eichenbaum, Rebelo, Trabandt (2022, JEDC): "Epidemics in the New Keynesian model"
% based on plot_by_type_results.m from the original Dynare replication files
% kindly provided by the authors (downloaded from Mathias Trabandt's homepage)
%-------------------------------------------------------------------------
% INPUTS
% - epidemic_type  [string]     type of epidemic, possible values:
%                               "demand", "supply", "both"
% - M_             [structure]  Dynare's model structure
% - oo_            [structure]  Dynare's results structure
%-------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

% common settings for plots
fsize = 12;
horz  = 100;
time  = 0:1:horz-1;

% title for figure
if epidemic_type == "demand"
    strtitle = "Response of consumption and hours when epidemic is a shock to demand";
elseif epidemic_type == "supply"
    strtitle = "Response of consumption and hours when epidemic is a shock to supply";
elseif epidemic_type == "both"
    strtitle = "Response of consumption and hours when epidemic is a shock to both demand and supply";
end
figure('name',strtitle,'units','normalized','outerposition',[0 0 0.75 0.75]);

% Consumption by Type
c_ss = oo_.steady_state(M_.endo_names=="c");
```

```
cs = oo_.endo_simul(M_.endo_names=="cs",:);
ci = oo_.endo_simul(M_.endo_names=="ci",:);
cr = oo_.endo_simul(M_.endo_names=="cr",:);
subplot(1,2,1);
hold on;
plot(time(1:end-1),100*(cs(2:horz)-c_ss)/c_ss,'r--','LineWidth',2);
plot(time(1:end-1),100*(ci(2:horz)-c_ss)/c_ss,'k-.','LineWidth',2);
plot(time(1:end-1),100*(cr(2:horz)-c_ss)/c_ss,'m:','LineWidth',2);
plot(time(1:end-1),0*time(1:end-1),'b-','LineWidth',1);
hold off
box off;
title('Consumption by Type','FontSize',fsize);
ylabel('% Dev. from Initial Steady State','FontSize',fsize);
xlabel('Weeks','FontSize',fsize);
set(gca,'FontSize',fsize);
legend('Susceptibles','Infected','Recovered','Location','SouthEast');
legend boxoff

% Hours by Type
n_ss = oo_.steady_state(M_.endo_names=="n");
ns = oo_.endo_simul(M_.endo_names=="ns",:);
ni = oo_.endo_simul(M_.endo_names=="ni",:);
nr = oo_.endo_simul(M_.endo_names=="nr",:);
subplot(1,2,2);
hold on;
plot(time(1:end-1),100*(ns(2:horz)-n_ss)/n_ss,'r--','LineWidth',2);
plot(time(1:end-1),100*(ni(2:horz)-n_ss)/n_ss,'k-.','LineWidth',2);
plot(time(1:end-1),100*(nr(2:horz)-n_ss)/n_ss,'m:','LineWidth',2);
plot(time(1:end-1),0*time(1:end-1),'b-','LineWidth',1);
hold off
box off;
title('Hours by Type','FontSize',fsize);
ylabel('% Dev. from Initial Steady State','FontSize',fsize);
xlabel('Weeks','FontSize',fsize);
set(gca,'FontSize',fsize);
legend('Susceptibles','Infected','Recovered','Location','best');
legend boxoff

if epidemic_type == "demand"
    suptitle("Figure 6: " + strtitle);
elseif epidemic_type == "supply"
    suptitle("Figure 8: " + strtitle);
elseif epidemic_type == "both"
    suptitle("Figure 10: " + strtitle);
end
```

A. Solutions

1 Solution to Case Study Eichenbaum, Rebelo and Trabandt (2022, JEDC): Epidemics in the New Keynesian model

2. Let's use Dynare's `#include "ert_model_sticky.mod"` directive to read in the sticky-price mod file into another mod file that just computes the steady-state. Note that during calibration all values for the pre-infection steady-state are already computed (with a subindex `_ss`), so we can simply put that into an `initval` block:

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_sticky_steady.mod

```
% ========================================================================
% Computes the steady-state of the sticky-price New Keynesian Model of
% Eichenbaum, Rebelo, and Trabandt (2022, JEDC): "Epidemics in the New
% Keynesian model".
% This illustrates using the include directive of Dynare's preprocessor.
% ------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

@#include "ert_model_sticky.mod"

%-------------------------------------------------------------------------%
% compute initial pre-infection steady-state
%-------------------------------------------------------------------------%
initval;
y = y_ss;
k = k_ss;
n = n_ss;
w = w_ss;
rk = rk_ss;
x = x_ss;
c = c_ss;
s = s_ss;
i = i_ss;
r = r_ss;
ns = ns_ss;
ni = ni_ss;
nr = nr_ss;
cs = cs_ss;
ci = ci_ss;
cr = cr_ss;
tau = tau_ss;
lambtilde = lambtilde_ss;
lamtau = lamtau_ss;
lami = lami_ss;
lams = lams_ss;
lamr = lamr_ss;
dd = dd_ss;
pop = pop_ss;
pbreve = pbreve_ss;
mc = mc_ss;
rr = rr_ss;
Rb = Rb_ss;
pie = pie_ss;
Kf = Kf_ss;
F = F_ss;
end;
resid;
steady;
```

3. In the paper the state variables capital, infected, susceptible, recovered, deceased and population have a "beginning-of-period" timing convention, whereas Dynare requires these variables to be "end-of-period". One could either lag all occurrences manually in the model equations, or use `"predetermined_variables"` such that Dynare's preprocessor does this transformation of timing. This is personal preference, but using `"predetermined_variables"` makes the equations appear more in line with the paper.

4. This is basically a copy & paste exercise combined with the `#include` directive. First, let's create a mod file for the flex-price economy called `"ert_model_flex.inc"`
   - Copy and paste the variable declarations and model equations of `ert_model_sticky.mod` into the new mod file.
   - Add a `_flex` to the variables to indicate that this is the same variable, but in the flex-price model. Similarly, adjust all appearances of the variable in the model equations. Hint: You might want to use MATLAB's *Find & Replace* feature for this.
   - Rename the parameter ξ in the flex-price model to ξ^f and set it equal to 0. Do so by either declaring a new parameter calibrated to 0 or using a *model local variable.*
   - Re-compute manually the steady-states of the auxiliary variables in the recursive pricing equations $K^{f,flex}$ and F^{flex}.

   The inc file might look like this:

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_flex.inc

```
% ========================================================================
% Stripped down flex-price version of New Keynesian Model of Eichenbaum,
% Rebelo, and Trabandt (2022, JEDC): "Epidemics in the New Keynesian model"
% based on the original Dynare replication files kindly provided by the
% authors (downloaded from Mathias Trabandt'sF homepage)
% ------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

%-------------------------------------------------------------------------%
% DECLARE ENDOGENOUS VARIABLES FOR THE FLEX-PRICE ECONOMY
%-------------------------------------------------------------------------%
var
y_flex          ${y^{flex}}$                (long_name='output (flex-price)')
k_flex          ${k^{flex}}$                (long_name='aggregate capital (flex-price)')
n_flex          ${n^{flex}}$                (long_name='aggregate labor (flex-price)')
w_flex          ${w^{flex}}$                (long_name='real wage (flex-price)')
rk_flex         ${r^{k,flex}}$              (long_name='real rental rate of capital (flex-price)')
x_flex          ${x^{flex}}$                (long_name='investment (flex-price)')
c_flex          ${c^{flex}}$                (long_name='aggregate consumption (flex-price)')
s_flex          ${s^{flex}}$                (long_name='susceptible (flex-price)')
i_flex          ${i^{flex}}$                (long_name='infected (flex-price)')
r_flex          ${r^{flex}}$                (long_name='recovered (flex-price)')
ns_flex         ${n^{s,flex}}$              (long_name='labor supply of susceptibles (flex-price)')
ni_flex         ${n^{i,flex}}$              (long_name='labor supply of infected (flex-price)')
nr_flex         ${n^{r,flex}}$              (long_name='labor supply of recovered (flex-price)')
cs_flex         ${c^{s,flex}}$              (long_name='consumption of susceptibles (flex-price)')
ci_flex         ${c^{i,flex}}$              (long_name='consumption of infected (flex-price)')
cr_flex         ${c^{r,flex}}$              (long_name='consumption of recovered (flex-price)')
tau_flex        ${\tau^{flex}}$             (long_name='newly infected (flex-price)')
lambtilde_flex  ${\tilde{\lambda}^{b,flex}}$ (long_name='Lagrange multiplier on household budget (flex-price)')
lamtau_flex     ${\lambda^{\tau,flex}}$     (long_name='Lagrange multiplier on transmission function (flex-price)')
lami_flex       ${\lambda^{i,flex}}$        (long_name='Lagrange multiplier on law of motion of infected (flex-price)')
lams_flex       ${\lambda^{s,flex}}$        (long_name='Lagrange multiplier on law of motion of susceptible (flex-price)')
lamr_flex       ${\lambda^{r,flex}}$        (long_name='Lagrange multiplier on law of motion of recovered (flex-price)')
dd_flex         ${d^{flex}}$                (long_name='deceased (flex-price)')
pop_flex        ${pop^{flex}}$              (long_name='population (flex-price)')
pbreve_flex     ${\breve{p}^{flex}}$        (long_name='inverse price dispersion (flex-price)')
mc_flex         ${mc^{flex}}$               (long_name='real marginal costs (flex-price)')
rr_flex         ${rr^{flex}}$               (long_name='real interest rate (flex-price)')
Rb_flex         ${R^{b,flex}}$              (long_name='nominal interest rate (flex-price)')
pie_flex        ${\pi^{flex}}$              (long_name='inflation (flex-price)')
Kf_flex         ${K^{f,flex}}$              (long_name='auxiliary variable 1 in recursive price setting (flex-price)')
F_flex          ${F^{flex}}$                (long_name='auxiliary variable 2 in recursive price setting (flex-price)')
;
```

```
47
48  %-----------------------------------------------------------------------%
49  % FLEX-PRICE MODEL EQUATIONS
50  %-----------------------------------------------------------------------%
51  % exactly the same as sticky-price equations but with added "_flex"
52  % notation and Calvo probability xi_flex set to 0
53
54  predetermined_variables k_flex i_flex s_flex r_flex dd_flex pop_flex;
55
56  model(differentiate_forward_vars);
57  #xi_flex = 0;
58  [name='F1) aggregate supply (flex-price)']
59  y_flex = pbreve_flex * A * k_flex^(1-alfa) * n_flex^alfa;
60  [name='F2) marginal costs (flex-price)']
61  mc_flex = w_flex^alfa * rk_flex^(1-alfa) / ( A * alfa^alfa * (1-alfa)^(1-alfa) );
62  [name='F3) optimal factor input (flex-price)']
63  w_flex = mc_flex * alfa * A * n_flex^(alfa-1) * k_flex^(1-alfa);
64  [name='F4) capital accumulation (flex-price)']
65  k_flex(+1) = (1-delta) * k_flex + x_flex;
66  [name='F5) aggregate demand (flex-price)']
67  y_flex = c_flex + x_flex + g_ss;
68  [name='F6) aggregate hours (flex-price)']
69  n_flex = (s_flex * ns_flex) + (i_flex * ni_flex) + (r_flex * nr_flex);
70  [name='F7) aggregate consumption (flex-price)']
71  c_flex = (s_flex * cs_flex) + (i_flex * ci_flex) + (r_flex * cr_flex);
72  [name='F8) transmission function for new infections (flex-price)']
73  tau_flex = pi1 * (s_flex * cs_flex) * (i_flex * ci_flex) + pi2 * (s_flex * ns_flex) * (i_flex * ni_flex) + pi3
        * (s_flex * i_flex);
74  [name='F9) law of motion susceptible population (flex-price)']
75  s_flex(+1) = s_flex - tau_flex;
76  [name='F10) law of motion infected population (flex-price)']
77  i_flex(+1) = i_flex + tau_flex - (pir + pid) * i_flex;
78  [name='F11) law of motion recovered population (flex-price)']
79  r_flex(+1) = r_flex + pir * i_flex;
80  [name='F12) law of motion deceased population (flex-price)']
81  dd_flex(+1) = dd_flex + pid * i_flex;
82  [name='F13) total population (flex-price)']
83  pop_flex(+1) = pop_flex - pid * i_flex;
84  [name='F14) first order condition wrt consumption susceptibles (flex-price)']
85  1 / cs_flex = lambtilde_flex - lamtau_flex * pi1 * (i_flex * ci_flex);
86  [name='F15) first order condition wrt consumption infected (flex-price)']
87  1 / ci_flex = lambtilde_flex;
88  [name='F16) first order condition wrt consumption recovered (flex-price)']
89  1 / cr_flex = lambtilde_flex;
90  [name='F17) first order condition wrt hours susceptibles (flex-price)']
91  theta * ns_flex = lambtilde_flex * w_flex + lamtau_flex * pi2 * (i_flex * ni_flex);
92  [name='F18) first order condition wrt hours infected (flex-price)']
93  theta * ni_flex = lambtilde_flex * w_flex;
94  [name='F19) first order condition wrt hours recovered (flex-price)']
95  theta * nr_flex = lambtilde_flex * w_flex;
96  [name='F20) first order condition wrt capital (flex-price)']
97  lambtilde_flex = betta * ( rk_flex(+1) + 1 - delta ) * lambtilde_flex(+1);
98  [name='F21) first order condition wrt newly infected (flex-price)']
99  lami_flex = lamtau_flex + lams_flex;
100 [name='F22) first order condition wrt susceptibles (flex-price)']
101 0 = log(cs_flex(+1)) - theta/2 * ns_flex(+1)^2
102   + lamtau_flex(+1) * ( pi1 * cs_flex(+1) * i_flex(+1) * ci_flex(+1)
103                       + pi2 * ns_flex(+1) * i_flex(+1) * ni_flex(+1)
104                       + pi3 * i_flex(+1)
105                       )
106   + lambtilde_flex(+1) * ( w_flex(+1) * ns_flex(+1) - cs_flex(+1) )
107   - lams_flex / betta + lams_flex(+1);
108 [name='F23) first order condition wrt infected (flex-price)']
109 0 = log(ci_flex(+1)) - theta/2 * ni_flex(+1)^2
110   + lambtilde_flex(+1) * ( w_flex(+1) * ni_flex(+1) - ci_flex(+1) )
111   - lami_flex/betta + lami_flex(+1) * (1 - pir - pid) + lamr_flex(+1)*pir;
112 [name='F24) first order condition wrt recovered (flex-price)']
113 0 = log(cr_flex(+1)) - theta/2 * nr_flex(+1)^2
114   + lambtilde_flex(+1) * ( w_flex(+1) * nr_flex(+1) - cr_flex(+1) )
115   - lamr_flex/betta + lamr_flex(+1);
116 [name='F25) first order condition wrt bonds (flex-price)']
117 lambtilde_flex = betta * rr_flex * lambtilde_flex(+1);
118 [name='F26) real interest rate']
119 rr_flex = Rb_flex / pie_flex(+1);
120 [name='F27) recursion nonlinear price setting 1 (flex-price)']
```

```
Kf_flex = gam * mc_flex * lambtilde_flex * y_flex + betta * xi_flex * pie_flex(+1)^(gam/(gam-1)) * Kf_flex(+1);
[name='F28) recursion nonlinear price setting 2 (flex-price)']
F_flex = lambtilde_flex * y_flex + betta * xi_flex * pie_flex(+1)^(1/(gam-1)) *F_flex(+1);
[name='F29) nonlinear price setting (flex-price)']
Kf_flex = F_flex * ( ( 1 - xi_flex * pie_flex^(1/(gam-1)) ) / (1 - xi_flex) )^(-(gam-1));
[name='F30) inverse price dispersion (flex-price)']
pbreve_flex^(-1) = (1-xi_flex) * ( ( 1 - xi_flex * pie_flex^(1/(gam-1)) ) / (1 - xi_flex) )^gam
                 + xi_flex * pie_flex^(gam/(gam-1)) / pbreve_flex(-1);
[name='F31) Taylor rule (flex-price)']
Rb_flex - STEADY_STATE(Rb_flex) = rpi * log( pie_flex / STEADY_STATE(pie_flex) ) + rx * log( y_flex / y_flex );
end;

%-----------------------------------------------------------------------%
% PRE-INFECTION STEADY-STATE
%-----------------------------------------------------------------------%
% recursion nonlinear price setting (flex-price)
Kf_flex_ss = 1/(1-betta*0 )*gam*mc_ss*lambtilde_ss*y_ss;
F_flex_ss  = 1/(1-betta*0 )*lambtilde_ss*y_ss;
```

As a mod file can contain several declarations of variables (`var` blocks) or model equations (`model` blocks), we can conveniently use `#include` to put both economies into the same mod file. So let's create a new mod file called `"ert_model_initval.mod"` and use `#include "ert_model_sticky.mod"` to read in the sticky-price mod file and similarly `#include "ert_model_flex.inc"` directive to read in the flex-price mod file. Then compute the pre-infection steady-state by using an `initval` block. Note that we need to set the macro directive
`#define FLEX_PRICE_OUTPUT_GAP = 1`
prior to including the flex-price model so the correct Taylor rule is chosen.

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_initval.mod

```
% ========================================================================
% Combines the sticky-price model equations and the flex-price model
% equations of the New Keynesian Model of Eichenbaum, Rebelo, and Trabandt
% (2022, JEDC): "Epidemics in the New Keynesian model" and uses an
% initval block to compute the steady-state.
% This also illustrates setting macro variables and the include directive
% of Dynare's preprocessor.
% ------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

@#define FLEX_PRICE_OUTPUT_GAP = 1
@#include "ert_model_sticky.mod"
@#include "ert_model_flex.inc"

%-----------------------------------------------------------------------%
% compute initial pre-infection steady-state
%-----------------------------------------------------------------------%
initval;
% most variables in sticky- and flex-price economies have same steady-state
y = y_ss;                   y_flex = y_ss;
k = k_ss;                   k_flex = k_ss;
n = n_ss;                   n_flex = n_ss;
w = w_ss;                   w_flex = w_ss;
x = x_ss;                   x_flex = x_ss;
c = c_ss;                   c_flex = c_ss;
s = s_ss;                   s_flex = s_ss;
i = i_ss;                   i_flex = i_ss;
r = r_ss;                   r_flex = r_ss;
rk = rk_ss;                 rk_flex = rk_ss;
ns = ns_ss;                 ns_flex = ns_ss;
ni = ni_ss;                 ni_flex = ni_ss;
nr = nr_ss;                 nr_flex = nr_ss;
cs = cs_ss;                 cs_flex = cs_ss;
ci = ci_ss;                 ci_flex = ci_ss;
cr = cr_ss;                 cr_flex = cr_ss;
dd = dd_ss;                 dd_flex = dd_ss;
mc = mc_ss;                 mc_flex = mc_ss;
rr = rr_ss;                 rr_flex = rr_ss;
Rb = Rb_ss;                 Rb_flex = Rb_ss;
tau = tau_ss;               tau_flex = tau_ss;
```

```
pop = pop_ss;              pop_flex = pop_ss;
pie = pie_ss;              pie_flex = pie_ss;
lami = lami_ss;            lami_flex = lami_ss;
lams = lams_ss;            lams_flex = lams_ss;
lamr = lamr_ss;            lamr_flex = lamr_ss;
lamtau = lamtau_ss;        lamtau_flex = lamtau_ss;
pbreve = pbreve_ss;        pbreve_flex = pbreve_ss;
lambtilde = lambtilde_ss;  lambtilde_flex = lambtilde_ss;
% recursive pricing variables have different steady-states
F = F_ss;                  F_flex = F_flex_ss;
Kf = Kf_ss;                Kf_flex = Kf_flex_ss;
end;
resid;
steady;
```

5. The key sentence in the manual interesting to our application is:

   > *Another one is to study how an economy, starting from arbitrary initial conditions at time 0 converges towards equilibrium. In this case models, the command histval permits to specify different historical initial values for variables with lags for the periods before the beginning of the simulation. Due to the design of Dynare, in this case initval is used to specify the terminal conditions.*

   For us the arbitrary initial condition is that infections break out at time 0 and we want to study how the economy converges towards equilibrium (which is not the pre-infection steady-state, see next exercise).

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_histval.mod

```
% ========================================================================
% Includes the scenario "The impact of an epidemic" into the New Keynesian
% Model of Eichenbaum, Rebelo, and Trabandt (2022, JEDC): "Epidemics in the
% New Keynesian model" by specifying an initial condition such that
% infections break out at time 0.
% This mod file also includes some pretty printing of the initial matrices
% created by Dynare.
% ------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

@#include "ert_model_initval.mod"

%-------------------------------------------------------------------------%
% use histval for initial condition that infections break out at time 0
%-------------------------------------------------------------------------%
histval;
% infections break out at time 0
i(0) = varepsilon;            i_flex(0) = varepsilon;
s(0) = 1-varepsilon;          s_flex(0) = 1-varepsilon;

% most variables set at pre-infection steady-state
y(0) = y_ss;                  y_flex(0) = y_ss;
k(0) = k_ss;                  k_flex(0) = k_ss;
n(0) = n_ss;                  n_flex(0) = n_ss;
w(0) = w_ss;                  w_flex(0) = w_ss;
x(0) = x_ss;                  x_flex(0) = x_ss;
c(0) = c_ss;                  c_flex(0) = c_ss;
r(0) = r_ss;                  r_flex(0) = r_ss;
rk(0) = rk_ss;                rk_flex(0) = rk_ss;
ns(0) = ns_ss;                ns_flex(0) = ns_ss;
ni(0) = ni_ss;                ni_flex(0) = ni_ss;
nr(0) = nr_ss;                nr_flex(0) = nr_ss;
cs(0) = cs_ss;                cs_flex(0) = cs_ss;
ci(0) = ci_ss;                ci_flex(0) = ci_ss;
cr(0) = cr_ss;                cr_flex(0) = cr_ss;
dd(0) = dd_ss;                dd_flex(0) = dd_ss;
mc(0) = mc_ss;                mc_flex(0) = mc_ss;
rr(0) = rr_ss;                rr_flex(0) = rr_ss;
Rb(0) = Rb_ss;                Rb_flex(0) = Rb_ss;
tau(0) = tau_ss;              tau_flex(0) = tau_ss;
pop(0) = pop_ss;              pop_flex(0) = pop_ss;
```

```
pie(0) = pie_ss;              pie_flex(0) = pie_ss;
lami(0) = lami_ss;            lami_flex(0) = lami_ss;
lams(0) = lams_ss;            lams_flex(0) = lams_ss;
lamr(0) = lamr_ss;            lamr_flex(0) = lamr_ss;
lamtau(0) = lamtau_ss;        lamtau_flex(0) = lamtau_ss;
pbreve(0) = pbreve_ss;        pbreve_flex(0) = pbreve_ss;
lambtilde(0) = lambtilde_ss;  lambtilde_flex(0) = lambtilde_ss;
% recursive pricing variables have different steady-states
F(0) = F_ss;                  F_flex(0) = F_flex_ss;
Kf(0) = Kf_ss;                Kf_flex(0) = Kf_flex_ss;
end;

perfect_foresight_setup(periods=500);

% have a look at how variables for perfect foresight algorithm are initialized
fprintf('\n\nINITIALIZATION OF PERFECT FORESIGHT ALGORITHM: oo_.steady_state\n')
disp(array2table(oo_.steady_state', 'VariableNames',M_.endo_names));
fprintf('\n\nINITIALIZATION OF PERFECT FORESIGHT ALGORITHM: M_.endo_histval\n')
disp(array2table(M_.endo_histval', 'VariableNames',M_.endo_names));
fprintf('\n\nINITIALIZATION OF PERFECT FORESIGHT ALGORITHM: oo_.endo_simul\n')
disp(array2table(oo_.endo_simul(:,[1:3 options_.periods+2])',...
                 'VariableNames',M_.endo_names,...
                 'RowNames',["t=0","t=1","t=2",("t="+string(options_.periods+2))]'));
fprintf('\n\nINITIALIZATION OF PERFECT FORESIGHT ALGORITHM: oo_.exo_simul\n')
oo_.exo_simul
```

6. The Newton algorithm requires the evaluation of the forward-looking variables at the terminal steady-state. But the terminal steady-state is an endogenous function of the epidemic dynamics; hence, it is not known a priori. Typically, this makes the perfect foresight solution algorithm inaccurate and often the solution cannot not found. But notice that the problem is just that we need to evaluate ONLY the forward-looking variables. So, one way out is to replace all forward-looking variables in the model by auxiliary variables that have a known terminal steady-state, such that the solution regains numerical accuracy. Dynare's `differentiate_forward_vars` option implements the following transformation automatically: Replace all (or a subset of) variables dated t+1, say X_{t+1}, by the sum of the current level and expected variation, X_t+dX_{t+1}, and introduce an auxiliary equation for the new variable $dX_t = X_t - X_{t-1}$. This transformation is done automatically by Dynare's preprocessor, which is a very good thing because doing such transformations manually is extremely error-prone.

7. Newton-type algorithms are very powerful if we are close to the actual solution as it belongs to the family of gradient-type optimizers. Whenever we want to study **large** deviations from an initial condition, Newton-type algorithms might have a hard time to find the solution. The idea of homotopy (also called divide-and-conquer) is therefore to subdivide the problem of finding the solution into smaller problems. So for example, if you have a large exogenous shock on impact of size 1, the idea is to first find a solution for a smaller shock size, say 0.1, and then use that as an initial condition to find a solution for a shock size of 0.2, 0.3,..., and finally 1. This is the default of Dynare.

   > *Whenever the maximum number of iterations is reached by the Newton algorithm, the perfect foresight solver uses a homotopy technique if it cannot solve the problem. Concretely, it divides the problem into smaller steps by diminishing the size of shocks and increasing them progressively until the problem converges.*

   But in our case we don't have a shock, but jump from no infections to infections. Particularly, we change the parameter π_3 from 0 to a large value, say 0.5, and this is a large jump. Try out the following mod file:

   progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_no_homotopy.mod

```
% ========================================================================
% Illustrates that the scenario "The impact of an epidemic" in the New
% Keynesian Model of Eichenbaum, Rebelo, and Trabandt (2022, JEDC):
% "Epidemics in the New Keynesian model" has convergence issues due to the
% large change in the infection probability of random meetings (pi3).
```

```
%------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

@#include "ert_model_histval.mod"

% use set_param_value to update the parameters in the M_ structure
set_param_value('pi1',2.568436063602094e-07);
set_param_value('pi2',1.593556989906377e-04);
set_param_value('pi3',0.499739472034583);

% run perfect foresight simulation (feel free to abort it using CTRL+C)
perfect_foresight_solver(maxit=100);
```

Note that even though Dynare has a built in switch to a homotopy method, it still fails, because we don't have a large shock or a large difference between initial and terminal conditions, but change parameters that affects the dynamics of the model to an endogenous terminal steady-state. Therefore, we need to implement the homotopy ourselves by looping over the parameter π_3. Try out the following mod file with homotopy:

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_homotopy.mod

```
% ========================================================================
% Illustrates homotopy over a parameter in a perfect foresight context.
% Particularly, this is required to simulate the scenario "The impact of an
% epidemic" in the New Keynesian Model of Eichenbaum, Rebelo, and Trabandt
% (2022, JEDC): "Epidemics in the New Keynesian model".
%------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

@#include "ert_model_histval.mod"

% changes in pi1 and pi2 are very small, so no need for homotopy
set_param_value('pi1',2.568436063602094e-07);
set_param_value('pi2',1.593556989906377e-04);

% homotopy over pi3 (otherwise the simulation does not converge)
pi3_final = 0.499739472034583;
pi3_steps = [pi3_final/3:0.02:pi3_final,pi3_final];
set_param_value('pi3',pi3_final/3); % set initial pi3 to one third of final value

% re-run perfect_foresight_solver for increasing values of pi3 taking
% previous simulation as initial guess for perfect foresight solver
for pi3_j = pi3_steps
    fprintf('pi3 = %f\n',pi3_j);
    set_param_value('pi3',pi3_j);
    perfect_foresight_solver(maxit=100);
end
```

8. This replicates Figures 5 and 6:

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_figures_demand.mod

```
% ========================================================================
% Replicates Figures 5 and 6 of
% Eichenbaum, Rebelo, and Trabandt (2022, JEDC):
% "Epidemics in the New Keynesian model"
%------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

@#include "ert_model_histval.mod"

% calibation targets for shares of pi-terms in T-function in SIR model
pi3_shr_target = 2/3;                 % share of T_0 jump due general infections
pi1_shr_target = (1-pi3_shr_target);  % share of T_0 jump due to consumption-based infections
pi2_shr_target = 0;                   % share of T_0 jump due to work-based infections
```

```
[pi1_final,pi2_final,pi3_final] = ert_model_go_calibrate_pi(250,varepsilon,pir,pid,pi1_shr_target,
    pi2_shr_target,RplusD_target,c_ss,n_ss);

% use set_param_value to update the parameters in the M_ structure
% changes in pi1 and pi2 are very small, so no need for homotopy
set_param_value('pi1',pi1_final);
set_param_value('pi2',pi2_final);

% homotopy over pi3 (otherwise the simulation does not converge)
pi3_steps = [pi3_final/3:0.02:pi3_final,pi3_final];
set_param_value('pi3',pi3_final/3); % set initial pi3 to one third of final value

% re-run perfect_foresight_solver for increasing values of pi3 taking
% previous simulation as initial guess for perfect foresight solver
for pi3_j = pi3_steps
    fprintf('pi3 = %f\n',pi3_j);
    set_param_value('pi3',pi3_j);
    perfect_foresight_solver(maxit=100);
end

% create figures
plot_also_flex_price_model = false;
ert_model_plot_agg_results("demand",M_,oo_,plot_also_flex_price_model);
ert_model_plot_by_type_results("demand",M_,oo_);
```

9. This replicates Figures 7 and 8:

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_figures_supply.mod

```
% ========================================================================
% Replicates Figures 7 and 8 of
% Eichenbaum, Rebelo, and Trabandt (2022, JEDC):
% "Epidemics in the New Keynesian model"
% ------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

@#include "ert_model_histval.mod"

% calibation targets for shares of pi-terms in T-function in SIR model
pi3_shr_target = 2/3;                 % share of T_0 jump due general infections
pi1_shr_target = 0;                   % share of T_0 jump due to consumption-based infections
pi2_shr_target = (1-pi3_shr_target);  % share of T_0 jump due to work-based infections
[pi1_final,pi2_final,pi3_final] = ert_model_go_calibrate_pi(250,varepsilon,pir,pid,pi1_shr_target,
    pi2_shr_target,RplusD_target,c_ss,n_ss);

% use set_param_value to update the parameters in the M_ structure
% changes in pi1 and pi2 are very small, so no need for homotopy
set_param_value('pi1',pi1_final);
set_param_value('pi2',pi2_final);

% homotopy over pi3 (otherwise the simulation does not converge)
pi3_steps = [pi3_final/3:0.02:pi3_final,pi3_final];
set_param_value('pi3',pi3_final/3); % set initial pi3 to one third of final value

% re-run perfect_foresight_solver for increasing values of pi3 taking
% previous simulation as initial guess for perfect foresight solver
for pi3_j = pi3_steps
    fprintf('pi3 = %f\n',pi3_j);
    set_param_value('pi3',pi3_j);
    perfect_foresight_solver(maxit=100);
end

% create figures
plot_also_flex_price_model = false;
ert_model_plot_agg_results("supply",M_,oo_,plot_also_flex_price_model);
ert_model_plot_by_type_results("supply",M_,oo_);
```

10. This replicates Figures 9 and 10:

progs/replications/Eichenbaum_Rebelo_Trabandt_2022/ert_model_figures_both.mod

```
% ========================================================================
% Replicates Figures 9 and 10 of
% Eichenbaum, Rebelo, and Trabandt (2022, JEDC):
% "Epidemics in the New Keynesian model"
%-------------------------------------------------------------------------
% Willi Mutschler (willi@mutschler.eu)
% Version: May 18, 2023
% ========================================================================

@#include "ert_model_histval.mod"

% calibation targets for shares of pi-terms in T-function in SIR model
pi3_shr_target = 2/3;                 % share of T_0 jump due general infections
pi1_shr_target = (1-pi3_shr_target)/2; % share of T_0 jump due to consumption-based infections
pi2_shr_target = (1-pi3_shr_target)/2; % share of T_0 jump due to work-based infections
[pi1_final,pi2_final,pi3_final] = ert_model_go_calibrate_pi(250,varepsilon,pir,pid,pi1_shr_target,
     pi2_shr_target,RplusD_target,c_ss,n_ss);

% use set_param_value to update the parameters in the M_ structure
% changes in pi1 and pi2 are very small, so no need for homotopy
set_param_value('pi1',pi1_final);
set_param_value('pi2',pi2_final);

% homotopy over pi3 (otherwise the simulation does not converge)
pi3_steps = [pi3_final/3:0.02:pi3_final,pi3_final];
set_param_value('pi3',pi3_final/3); % set initial pi3 to one third of final value

% re-run perfect_foresight_solver for increasing values of pi3 taking
% previous simulation as initial guess for perfect foresight solver
for pi3_j = pi3_steps
    fprintf('pi3 = %f\n',pi3_j);
    set_param_value('pi3',pi3_j);
    perfect_foresight_solver(maxit=100);
end

% create figures
plot_also_flex_price_model = true;
ert_model_plot_agg_results("both",M_,oo_,plot_also_flex_price_model);
ert_model_plot_by_type_results("both",M_,oo_);

dynasave('willi.txt');
```