

Examine the FRB/US EDO Model

Gary Young

June 25, 2016

Contents

1	Introduction	5
	Appendices	7
A	Original Files	9
A.1	Dynare_edo_mod	9
A.2	Dynare_edo_steadystate.m	18
A.3	linearized.mod	24
A.4	linearized_steadystate.m	32
A.5	readme.txt	38
B	Notes, Bibliography and Indexes	39
B.1	Chunks	39
B.2	Index	39

Chapter 1

Introduction

Appendices

Appendix A

Original Files

A.1 Dynare_edo_mod

9 $\langle \text{srcedo}/\text{Dynare.edo.mod } 9 \rangle \equiv$

```
var RC RK WC WK YC YK MCC MCK KC KK PKB R L QK HC HSC HK HSK UHC UHSC UHK UHSK empC HrC e
DIFFREALGDP_obs DIFFREALEC_obs DIFFREALEIK_obs DIFFREALECD_obs DIFFREALECH_obs DIFFREALW_obs A
```

```
varexo eHG eXiL eLpref eR eMUZK eMUZM ePMKC ePMKK eEFFECH eEFFECD eEFFK eB eSTAR;
```

```
parameters
```

```
h r_inf r_y r_dy phi_pc phi_H phi_wc phi_ic phi_cd phi_ech gam_pc gam_wc gam_ic gam_icd rho_R r
rho_EFFECD rho_HG rho_EFFECH tp2 ONE MUZMSS MUZKSS r_dinf rpr phi_u rho_MUZK rho_MUZM pbeta de
theta_k theta_wc theta_wk g_y a_ks s_AS gam_h gam_ech s_k s_ecdc eta_cnn eta_cd eta_ch
icoef mu_ betarl MUZCSS RCSS RKSS WCSS WKSS YCSS YKSS MCCSS MCKSS KCSS KKSS LSS HCSS HKSS QKSS
MUCSS MUKSS AHSS ECDSS KCDSS QCDSS RCDSS ECHSS KCHSS QCHSS RCHSS UKSS UCSS USS MUKSShabit MUCSS
INFCNASS INFCORSS INFC10SS RT2SS beta_0 beta_2 beta_ PYSS AA DD RR
eta_cd_eta_cnn eta_ch_eta_cnn Rnr ycbi_ykb hc_hk HSS ycbi ykb YYSS s_k_ecd s_c_ech s_k_eik s_yc
sig_HG sig_XiL sig_lpref sig_R sig_MUZK sig_MUZM sig_PMKC sig_PMKK sig_EFFECH sig_EFFECD sig_E
HSKSS HSCSS HRCSS HrKSS A_HC sigman sigmah A_HK xsi_NC xsi_HrC xsi_NK xsi_HrK rho_XiL rho_lpref
empCSS empKSS HrSKSS HrSCSS empSCSS empSKSS UHCSS UHKSS UHSCSS UHSKSS unempSS DIFFREALGDPSS DIF
DIFFREALECHSS DIFFREALEIKSS DIFFREALWSS RL1SS RL2SS RL3SS RL4SS RL5SS
RL6SS RL7SS DIFFREALGDPSS_obs DIFFREALECSS_obs DIFFREALEIKSS_obs DIFFREALECDSS_obs
DIFFREALECHSS_obs DIFFREALWSS_obs INFCNASS_obs INFCORSS_obs INFKSS_obs
RSS_obs RT2SS_obs unempSS_obs;
```

```
//estimated_params;
```

```
h = 0.715162417869797;
```

```

r_inf      = 1.46344163969035;
r_y        = 0.263123294207851;
phi_pc     = 3.54471453295450;
phi_H      = 3.22894079106560;
phi_wc     = 5.49395755514723;
phi_ic     = 0.253308786976374;
phi_cd     = 0.470089385005009;
phi_ech    = 9.13986886546163;
gam_pc     = 0.314488926051065;
gam_wc     = -0.230018833252054;
sigman     = 39.4075260618789;
sigmah     = 21.8859803402692;
rho_R      = 0.833200065745674;
rho_XiL    = 0.263567746111198;
rho_lpref  = 0.979092048897712;
rho_B      = 0.895267027146152;
rho_STAR   = 0.909187927454138;
rho_EFFK   = 0.937829274540004;
rho_EFFECD = -0.240286975088701;
rho_HG     = 0.582395471123139;
rho_EFFECH = 0.877235725078934;
tp2        = 0.000307314910763576;
sig_HG     = 0.579315931803017;
sig_XiL    = 2.49313873916751;
sig_lpref  = 5.66476748114241;
sig_R      = 0.124100461010359;
sig_MUZK   = 0.936167718269030;
sig_MUZM   = 0.597390920898135;
sig_PMKC   = 0.451830653200989;
sig_PMKK   = 0.685376191952156;
sig_EFFECH = 0.514704527091087;
sig_EFFECD = 9.11199585973990;
sig_EFFK   = 0.402779878811407;
sig_B      = 0.295232712196573;
sig_STAR   = 0.104877885500673;
//end_estimated_params;

//calibrated_params;
r_dy = 0;
ONE = 1;
MUZKSS = 1.009250;
MUZMSS = 1.001000;
gam_ic = 1.0;
gam_icd = 1.0;
r_dinf = 0;
rpr = 0.965;

```

```
phi_u = 1;
rho_MUZK = 0;
rho_MUZM = 0;
pbeta = 0.99862;
delta_ = 0.03;
h_cd = 0.0;
h_ch = 0.0;
delta_cd = 0.055;
delta_ch = 0.0035;
alpha_ = 0.26;
theta_c = 7;
theta_k = 7;
unempSS = .06;
g_y = 0.0;
a_ks = 0.2;
s_AS = 0.2;
gam_h = 1;
gam_ech = 1;
icoef = 3;
betarl = .958;
//end_calibrated_params;

//free_params;
//A_HC;
//A_HK;
//xsi_NC;
//xsi_HrC;
//xsi_NK;
//xsi_HrK;
//theta_wc;
//theta_wk;
//infkbar;
//infcbars;
//infwcbar;
//infwkbar;
//Pybar;
//Yybar;
//mu_yc;
//mu_yk;
//s_k;
//s_ecdc;
//eta_cnn;
//eta_cd;
//eta_ch;
//mu_;
//end_free_params;
```

```
//calibrated ME
```

```
//*****  
//MODEL BLOCK  
//*****
```

```
model;  
RC-MCC*YC/UC/KC(-1)*alpha_*MUK=0;  
RK-MCK*YK/UK/KK(-1)*alpha_*MUK=0;  
WC-MCC*YC/HC*(1-alpha_)=0;  
WK-MCK*YK/HK*(1-alpha_)=0;  
YC-(UC*KC(-1)/MUK)^alpha_*(HC)^(1-alpha_)=0;  
YK-(UK*KK(-1)/MUK)^alpha_*(HK)^(1-alpha_)=0;  
MCC*YC*theta_c-(theta_c-1)*YC-100*phi_pc*(INFC-gam_pc*INFC(-1)-(1-gam_pc)*INFCSS)*INFC  
MCK*YK*theta_k/PKB-(theta_k-1)*YK-100*phi_pc*(INFK-gam_pc*INFK(-1)-(1-gam_pc)*INFKSS)  
QK-beta_*(1/EFFK)*(((1-delta_)*QK(+1)+RC(+1)*UC(+1))*L(+1)/MUK(+1)/L)=0;  
QK-beta_*(1/EFFK)*(((1-delta_)*QK(+1)+RK(+1)*UK(+1))*L(+1)/MUK(+1)/L)=0;  
L-betas*R/rpr/INFC(+1)/MUC(+1)*L(+1)=0;  
ln(R/RSS)-rho_R*ln(R(-1)/RSS)-(1-rho_R)*(r_inf*ln(INFCNA/INFCNASS)+r_dinf*(ln(INFCNA/INFCNASS)  
L-eta_cnn/(EC-h*EC(-1)/MUC)+eta_cnn*beta_*h/(MUC(+1)*EC(+1)-h*EC)=0;  
KK-(1-delta_)*KK(-1)/MUK+KC-(1-delta_)*KC(-1)/MUK-1*EIK+mu_*((UK^(1+1/phi_u)-1)/(1+1/phi_u))=0;  
  
// XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
// labor block  
// TOTAL LABOR INPUT (called "L" in the paper, I kept the "H" notation of the original paper)  
-100+UHC*theta_wc-(theta_wc-1)*WC-100*phi_wc*(INFWC-gam_wc*INFWC(-1)-(1-gam_wc)*INFWCSS)*INFWC  
UWSC*theta_h/10*(HSC/HSK-gam_h*HSC(-1)/HSK(-1)-(1-gam_h)*HSCSS/HSKSS);//+100*exiL=0;  
-100+UHK*theta_wk-(theta_wk-1)*WK-100*phi_wc*(INFWK-gam_wc*INFWK(-1)-(1-gam_wc)*INFWCSS)*INFWK  
UWSC*theta_h/10*(HSC/HSK-gam_h*HSC(-1)/HSK(-1)-(1-gam_h)*HSCSS/HSKSS);//+100*exiL=0;  
UHC*L*Lpref-A_HC*((1+sigman)/(1+sigman/(1+sigmah)))*(HC)^(-1+(1+sigman)/(1+sigman/(1+sigmah))))  
UWSC*L*Lpref-A_HC*((1+sigman)/(1+sigman/(1+sigmah)))*(HSC)^(-1+(1+sigman)/(1+sigman/(1+sigmah))))  
UHK*L*Lpref-A_HK*((1+sigman)/(1+sigman/(1+sigmah)))*(HK)^(-1+(1+sigman)/(1+sigman/(1+sigmah))))  
UWSC*L*Lpref-A_HK*((1+sigman)/(1+sigman/(1+sigmah)))*(HSK)^(-1+(1+sigman)/(1+sigman/(1+sigmah))))  
empC-((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(-1/(1+sigmah+sigman))*HC^(1/(1+sigman/(1+sigmah)))  
HrC-((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(1/(1+sigmah))*empC^(sigman/(1+sigmah))=0;  
empK-((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(-1/(1+sigmah+sigman))*HK^(1/(1+sigman/(1+sigmah)))  
HrK-((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(1/(1+sigmah))*empK^(sigman/(1+sigmah))=0;  
empSC-((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(-1/(1+sigmah+sigman))*HSC^(1/(1+sigman/(1+sigmah)))  
HrSC-((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(1/(1+sigmah))*empSC^(sigman/(1+sigmah))=0;  
empSK-((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(-1/(1+sigmah+sigman))*HSK^(1/(1+sigman/(1+sigmah)))  
HrSK-((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(1/(1+sigmah))*empSK^(sigman/(1+sigmah))=0;  
unemp-(empSC+empSK-(empC+empK))/(empSC+empSK)=0;  
PKB-(1-100*phi_ic*(EIK-gam_ic*EIK(-1)-(1-gam_ic)*EIKSS)/(KC(-1)+KK(-1))*MUK)*QK-beta_  
YC-EC-ECH-0.2*YC*SS*HG=0;
```

```

ln(INFWC)-ln(WC)+ln(WC(-1))-ln(MUC)-ln(INFC)=0;
ln(INFWK)-ln(WK)+ln(WK(-1))-ln(MUC)-ln(INFC)=0;
ln(INFK)-ln(INFC)-ln(PKB)+ln(PKB(-1))+ln(MUK)-ln(MUC)=0;
YK-EIK-ECD-0.2*YKSS*HG=0;
ln(DIFFNORMGDP)-(1-s_k)*(ln(YC)-ln(YC(-1)))-s_k*(ln(YK)-ln(YK(-1)))=0;
ln(NORMINFGDP)-s_k*(ln(PKB)-ln(PKB(-1)))=0;
ln(DIFFREALGDP)-ln(DIFFNORMGDP)-(1-s_k)*ln(MUC)-s_k*ln(MUK)=0;
ln(DIFFREALEC)-ln(EC)+ln(EC(-1))-ln(MUC)=0;
ln(DIFFREALEIK)-ln(EIK)+ln(EIK(-1))-ln(MUK)=0;

// Identities
ln(DIFFREALW)-HCSS/AHSS*(ln(INFWC))-HKSS/AHSS*(ln(INFWK))+ln(INFC)=0;

// XXXXXXXXXXXXXXXXXXXXXXXX
// Aggregate hours equals agg hours in each sector
AH-HC-HK=0;
ln(INFGDP)-ln(INFC)-ln(YC*MUC/YC(-1))+ln(DIFFREALGDP)-ln((1+PKB*YK/YC)/(1+PKB(-1)*YK(-1)/YC(-1)))=0;
ln(INFCNA)-(1-s_ecdc)*ln(INFC)-s_ecdc*ln(INFK)=0;
ln(INFCOR)-(1-s_ecdc)*ln(INFC)-s_ecdc*ln(INFK)=0;
ln(GAP)-(1-s_k)*ln(YC/YCSS)-s_k*ln(YK/YKSS)=0;
ln(PFGAP)-(1-alpha_)*((1-s_k)*ln(HC/HCSS)+s_k*ln(HK/HKSS))-alpha_*((1-s_k)*ln(UC/USS)+s_k*ln(UK/USS))=0;
ln(INFC10)-betarl*ln(INFC10(+1))-(1-betarl)*ln(INFCOR)=0;

// See Section 8: Data Identities

// new equations
// Durable Block

KD-(1-delta_cd)*KD(-1)/MUK-ECD=0;
L*RCD-eta_cd/(KD(-1)/MUK-h_cd*LAGKD(-1)/(MUK(-1)*MUK))+beta_*eta_cd*h_cd/(KD-h_cd*KD(-1)/MUK)=0;
QCD-beta_*(1/EFECD)*L(+1)/L/MUK(+1)*(RCD(+1)+(1-delta_cd)*QCD(+1))=0;
PKB-QCD*(1-100*phi_cd*(ECD-gam_icd*ECD(-1)-(1-gam_icd)*ECDSS)/KD(-1)*MUK) - beta_*(1/EFECD)*100*phi_cd*(ECD-gam_icd*ECD(-1)-(1-gam_icd)*ECDSS)/KD(-1)*MUK=0;

// Housing Block
L*RCH-eta_ch/(KCH(-1)/MUC-h_ch*LAGKCH(-1)/(MUC*MUC(-1)))+beta_*eta_ch*h_ch/(KCH-h_ch*KCH(-1)/MUC)=0;
QCH-beta_*(1/EFECH)*L(+1)/L/MUC(+1)*(RCH(+1)+(1-delta_ch)*QCH(+1))=0;
1*ECH+(1-delta_ch)*KCH(-1)/MUC-KCH=0;
1-QCH*(1-100*phi_ech*(ECH-gam_ech*ECH(-1)-(1-gam_ech)*ECHSS)/KCH(-1)*MUC) - beta_*(1/EFECH)*100*phi_ech*(ECH-gam_ech*ECH(-1)-(1-gam_ech)*ECHSS)/KCH(-1)*MUC=0;
ln(KD(-1))-ln(LAGKD)=0;
ln(KCH(-1))-ln(LAGKCH)=0;
RK-QK*mu_*UK^(1/phi_u)=0;
RC-QK*mu_*UC^(1/phi_u)=0;
ln(DIFFREALECH)-ln(MUC)-ln(ECH)+ln(ECH(-1))=0;
ln(DIFFREALECD)-ln(MUK)-ln(ECD)+ln(ECD(-1))=0;
ln(betas/beta_-rho_B*ln(betas(-1)/beta_-eB)=0;

```

```

ln(XiL)-rho_XiL*ln(XiL(-1))-eXiL=0;
ln(Lpref)-rho_lpref*ln(Lpref(-1))-eLpref=0;
ln(EFFK)-rho_EFFK*ln(EFFK(-1))-eEFFK=0;
ln(MUZK/MUZKSS)-eMUZK=0;
ln(MUZM/MUZMSS)-eMUZM=0;
ln(HG)-rho_HG*ln(HG(-1))-eHG=0;
ln(MUC)-ln(MUZM)-alpha_*ln(MUZK)=0;
ln(MUK)-ln(MUZM)-ln(MUZK)=0;
ln(EFFECD)-rho_EFFECD*ln(EFFECD(-1))-eEFFECD=0;
ln(EFFECH)-rho_EFFECH*ln(EFFECH(-1))-eEFFECH=0;
ln(STAR)-rho_STAR*ln(STAR(-1))-eSTAR=0;
ln(RL1) - ln(R(+1))=0;
ln(RL2) - ln(RL1(+1))=0;
ln(RL3) - ln(RL2(+1))=0;
ln(RL4) - ln(RL3(+1))=0;
ln(RL5) - ln(RL4(+1))=0;
ln(RL6) - ln(RL5(+1))=0;
ln(RL7) - ln(RL6(+1))=0;
ln(RT2) - tp2 - 0.125*(ln(R) + ln(RL1) + ln(RL2) + ln(RL3) + ln(RL4) + ln(RL5) + ln(RL6) + ln(RL7))=0;

//measurement_equations;
ln(DIFFREALGDP_obs/DIFFREALGDPSS_obs) = ln(DIFFREALGDP/DIFFREALGDPSS);
ln(DIFFREALEC_obs/DIFFREALECSS_obs)   = ln(DIFFREALEC/DIFFREALECSS);
ln(DIFFREALEIK_obs/DIFFREALEIKSS_obs)  = ln(DIFFREALEIK/DIFFREALEIKSS);
ln(DIFFREALECD_obs/DIFFREALECDSS_obs)  = ln(DIFFREALECD/DIFFREALECDSS);
ln(DIFFREALECH_obs/DIFFREALECHSS_obs)  = ln(DIFFREALECH/DIFFREALECHSS);
ln(DIFFREALW_obs/DIFFREALWSS_obs)      = ln(DIFFREALW/DIFFREALWSS);
ln(AH_obs)                             = ln(AH/AHSS);
ln(INFCNA_obs/INFCNASS_obs)             = ln(INFCNA/INFCNASS);
ln(INFCOR_obs/INFCORSS_obs)            = ln(INFCOR/INFCORSS);
ln(INFK_obs/INFKSS_obs)                 = ln(INFK/INFKSS);
ln(R_obs/RSS_obs)                      = ln(R/RSS);
ln(RT2_obs/RT2SS_obs)                  = ln(RT2/RT2SS);
ln(unemp_obs/unempSS_obs)              = ln(unemp/unempSS);
//end_measurement_equations;
end;

varobs DIFFREALGDP_obs DIFFREALEC_obs DIFFREALEIK_obs DIFFREALECD_obs DIFFREALECH_obs;

shocks;
var eHG;
stderr sig_HG;
var eXiL;
stderr sig_XiL;
var eLpref;
stderr sig_lpref;

```

```
var eR;
stderr sig_R;
var eMUZK;
stderr sig_MUZK;
var eMUZM;
stderr sig_MUZM;
var ePMKC;
stderr sig_PMKC;
var ePMKK;
stderr sig_PMKK;
var eEFFECH;
stderr sig_EFFECH;
var eEFFECD;
stderr sig_EFFECD;
var eEFFK;
stderr sig_EFFK;
var eB;
stderr sig_B;
var eSTAR;
stderr sig_STAR;
```

```
var DIFFREALGDP_obs;
stderr 0.3;
var DIFFFREALEC_obs;
stderr 0.1;
var DIFFFREALEIK_obs;
stderr 1.5;
var DIFFFREALECD_obs;
stderr 1.5;
var DIFFFREALECH_obs;
stderr 1.5;
var DIFFREALW_obs;
stderr 0.3;
var AH_obs;
stderr 0.3;
var INFCNA_obs;
stderr 0.5;
var INFCOR_obs;
stderr 0.05;
var INFK_obs;
stderr 0.2;
var RT2_obs;
stderr 0.1;
var unemp_obs;
stderr 4;
```

end;

steady;

estimated_params;

h	, .673	, -1	, 1	, uniform_pdf	, , , -1
r_inf	, 1.461	, -999	, 999	, normal_pdf	, 1.5000
r_y	, 0.214	, -999	, 999	, normal_pdf	, 0.125
phi_pc	, 3.126	, 0	, 999	, gamma_pdf	, 4.0000
phi_H	, 4.064	, 0	, 999	, gamma_pdf	, 4.0000
phi_wc	, 5.119	, 0	, 999	, gamma_pdf	, 4.0000
phi_ic	, .325	, 0	, 999	, gamma_pdf	, 4.0000
phi_cd	, .651	, 0	, 999	, gamma_pdf	, 4.0000
phi_ech	, 10.948	, 0	, 999	, gamma_pdf	, 4.0000
gam_pc	, 0.386	, -999	, 999	, normal_pdf	, 0.000
gam_wc	, 0.213	, -999	, 999	, normal_pdf	, 0.000
sigman	, 1.25	, 0	, 999	, gamma_pdf	, 1.25
sigmah	, 10	, 0	, 999	, gamma_pdf	, 10
rho_R	, 0.654	, -1	, 1	, normal_pdf	, 0.5
rho_XiL	, 0.654	, -1	, 1	, normal_pdf	, 0.5
rho_lpref	, 0.954	, -1	, 1	, normal_pdf	, 0.5
rho_B	, 0.825	, -1	, 1	, normal_pdf	, 0
rho_STAR	, 0.825	, -1	, 1	, normal_pdf	, 0
rho_EFFK	, 0.850	, -1	, 1	, normal_pdf	, 0
rho_EFFECD	, .230	, -1	, 1	, normal_pdf	, 0
rho_HG	, 0.596	, 0	, 1	, beta_pdf	, 0.5
rho_EFFECH	, 0.844	, -1	, 1	, normal_pdf	, 0
tp2	, 0.001	, -999	, 999	, normal_pdf	, 0.0

stderr eHG	, .745	, 0.0001	, 999	, inv_gamma_pdf	, 1.772454
stderr eXiL	, 3.621	, 0.0001	, 999	, inv_gamma_pdf	, 1.772454
stderr eLpref	, 1.621	, 0.0001	, 999	, inv_gamma_pdf	, 1.772454
stderr eR	, 0.165	, 0.0001	, 999	, inv_gamma_pdf	, 0.354491
stderr eMUZK	, .834	, 0.0001	, 999	, inv_gamma_pdf	, 0.443113
stderr eMUZM	, .484	, 0.0001	, 999	, inv_gamma_pdf	, 0.443113
stderr ePMKC	, .391	, 0.0001	, 999	, inv_gamma_pdf	, 0.354491
stderr ePMKK	, .552	, 0.0001	, 999	, inv_gamma_pdf	, 0.354491
stderr eEFFECH	, .526	, 0.0001	, 999	, inv_gamma_pdf	, 1.772454
stderr eEFFECD	, 13.349	, 0.0001	, 999	, inv_gamma_pdf	, 1.772454
stderr eEFFK	, .499	, 0.0001	, 999	, inv_gamma_pdf	, 1.772454
stderr eB	, 0.5	, 0.0001	, 999	, inv_gamma_pdf	, 1.772454
stderr eSTAR	, 0.05	, 0.0001	, 999	, inv_gamma_pdf	, 0.354491

end;

options_.order = 1;


```
options_.jacobian_flag = 1;  
options_.nonlin = 1;
```

```
stoch_simul(order=1,irf=40,nograph);
```

This code is written to file `srcedo/Dynare.edo.mod`.

A.2 Dynare_edo_steadystate.m

```

18  <srcedo/Dynare.edo.steadystate.m 18>≡
    function [ys,check] = unlinearized_edo_steadystate(ys,exe)
        global M_

        check = 0;

        NumberofParameters=M_.param_nbr;
        for i=1:NumberofParameters
            paramname=deblank(M_.param_names(i,:));
            eval([paramname ' =M_.params(' int2str(i) ');']);
        end;

        %start_steady_state;

        beta_0 = pbeta;
        beta_2 = pbeta*rpr; % s.s. funds rate premium
        beta_ = beta_2;
        MUZCSS=1;
        ONE=1;
        USS=1;
        MUKSS=MUKSS*MUZMSS;
        MUCSS=MUKSS^alpha_*MUZMSS;
        MUKSShabit=MUKSS;
        MUCSShabit=MUCSS;
        PKBSS=theta_k/(theta_k-1)*(theta_c-1)/theta_c;
        PYSS=1;
        MCCSS=(theta_c-1)/theta_c;
        MCKSS=(theta_k-1)/theta_k;
        RKSS=MUKSS/beta_2-(1-delta_);
        RCSS=MUKSS/beta_2-(1-delta_);
        RCHSS=MUCSS/beta_2-(1-delta_ch); % Housing sector
        RCDSS=MUKSS/beta_2-(1-delta_cd); % Durable sector
        USS=1;
        mu_=RCSS;
        AA=alpha_/RKSS*MCKSS;
        DD = 0.135;
        RR = 0.075;
        eta_cnn=1;
        eta_cd_eta_cnn=DD/((MUKSShabit-beta_2*h_cd)/(1-beta_2*h/MUCSShabit)*(1-h/MUCSShabit));
        eta_ch_eta_cnn=RR/((MUCSShabit-beta_2*h_ch)/(1-beta_2*h/MUCSShabit)*(1-h/MUCSShabit));
        eta_ch=eta_ch_eta_cnn;
        eta_cd=eta_cd_eta_cnn;
        DD=eta_cd_eta_cnn*(MUKSShabit-beta_2*h_cd)/(1-beta_2*h/MUCSShabit)*(1-h/MUCSShabit);
        RR=eta_ch_eta_cnn*(MUCSShabit-beta_2*h_ch)/(1-beta_2*h/MUCSShabit)*(1-h/MUCSShabit);

```

```

Rnr=(1-(1-delta_)/MUKSS)*AA*MUKSS;
ycbi_ykb=((1-s_AS)-Rnr)/((DD*(1-s_AS)/(1+RR))+Rnr);
hc_hk=ycbi_ykb*(RCSS*MCKSS/(RKSS*MCCSS))^(alpha_/(1-alpha_));
HSS=0.25;
AHSS=HSS;
HKSS=HSS/(1+hc_hk);
HCSS=HSS-HKSS;
HrCSS=1/3;
HrKSS=1/3;
empCSS=HCSS/HrCSS;
empKSS=HKSS/HrKSS;
ycbi=HCSS*(AA)^(alpha_/(1-alpha_));
ykb=HKSS*(AA)^(alpha_/(1-alpha_));
YCSS=ycbi;
YKSS=ykb;
KCSS=AA*ycbi*MUKSS;
KKSS=AA*ykb*MUKSS;
ECHSS=RR/(1+RR)*ycbi*(1-s_AS);
ECSS=1/(1+RR)*ycbi*(1-s_AS);
ECDSS=DD*PKBSS*ECSS;
EIKSS=(1-(1-delta_)/MUKSS)*(KCSS+KKSS);
KCDSS=ECDSS/(1-(1-delta_cd)/MUKSS);
KCHSS=ECHSS/(1-(1-delta_ch)/MUCSS);
YYSS=(YCSS+YKSS*PKBSS)/PYSS;
s_k_ecd=ECDSS/YKSS;
s_c_ech=ECHSS/YCSS;
s_k_eik=EIKSS/YKSS;
s_yc = (YCSS/YYSS);
s_ecdc=PKBSS*ECDSS/(ECSS+PKBSS*ECDSS+(MUCSS/beta_2-1+delta_ch)*KCHSS);
INFCNASS=exp(.02/4);
INFCSS = INFCNASS*((MUZCSS/MUZKSS)^(1-alpha_))^(s_ecdc);
INFCORSS=INFCNASS;
INFKSS=INFCSS*(MUZCSS/MUZKSS)^(1-alpha_);
INFWCSS=INFCSS*MUZKSS^alpha_*MUZMSS;
INFWKSS=INFWCSS;
RSS=INFCSS/beta_0*MUCSS;
RT2SS=exp(tp2)*RSS;
INFC10SS = INFCNASS;
IMPHSSS = RCHSS*KCHSS;
s_k=PKBSS*YKSS/YYSS;
INFGDPSS=INFCSS*(YCSS/YYSS)*INFKSS*(YKSS*PKBSS/(YYSS));
LSS=eta_cnn/(ECSS*(1-h/MUCSShabit))-eta_cnn*beta_2*h/(ECSS*(MUCSShabit-h));
WCSS=MCCSS*(1-alpha_)*YCSS/HCSS;
WKSS=MCKSS*(1-alpha_)*YKSS/HKSS;
xsiN_xsiH_C = ((HrCSS/empCSS)^(1+sigmah))/(1+1/sigmah);
xsiN_xsiH_K = ((HrKSS/empKSS)^(1+sigmah))/(1+1/sigmah);

```

```

gC = (1/(1+sigman) + 1/sigmah)*(xsiN_xsiH_C*(1+sigmah)/sigmah)^(-(1+sigman)/(1+sigman));
markup_xsiN_C = (HCSS^((1+sigmah)*(1+sigman)/(1+sigmah+sigman)-1))*gC/(LSS*WCSS);
gK = (1/(1+sigman) + 1/sigmah)*(xsiN_xsiH_K*(1+sigmah)/sigmah)^(-(1+sigman)/(1+sigman));
markup_xsiN_K = (HKSS^((1+sigmah)*(1+sigman)/(1+sigmah+sigman)-1))*gK/(LSS*WKSS);
markup_w = (1-unempSS)^((1+sigmah+sigman)/(1+sigmah) - 1 - sigman);
theta_wc = markup_w/(markup_w - 1); theta_wk = theta_wc;
A_HC=LSS*(theta_wc-1)/theta_wc*WCSS/(((1+sigman)/(1+sigman)/(1+sigmah)))*HCSS^(-1+(1+sigman));
A_HK=LSS*(theta_wk-1)/theta_wk*WKSS/(((1+sigman)/(1+sigman)/(1+sigmah)))*HKSS^(-1+(1+sigman));
xsi_NC=A_HC/((1/(1+sigman)+1/sigmah)*(HCSS^sigman/HrCSS^(1+sigman+sigmah)))^((1+sigman)/(1+sigman));
xsi_NK=A_HK/((1/(1+sigman)+1/sigmah)*(HKSS^sigman/HrKSS^(1+sigman+sigmah)))^((1+sigman)/(1+sigman));
xsi_HrC=xsi_NC*(1+sigmah)/sigmah*(HCSS^sigman/HrCSS^(1+sigman+sigmah));
xsi_HrK=xsi_NK*(1+sigmah)/sigmah*(HKSS^sigman/HrKSS^(1+sigman+sigmah));
UHCSS=A_HC*((1+sigman)/(1+sigman/(1+sigmah)))*HCSS^(-1+(1+sigman)/(1+sigman/(1+sigmah)));
UHKSS=A_HK*((1+sigman)/(1+sigman/(1+sigmah)))*HKSS^(-1+(1+sigman)/(1+sigman/(1+sigmah)));
HSCSS=(WCSS*LSS/(A_HC*((1+sigman)/(1+sigman/(1+sigmah))))^(1/(-1+(1+sigman)/(1+sigman/(1+sigmah)))));
HSKSS=(WKSS*LSS/(A_HK*((1+sigman)/(1+sigman/(1+sigmah))))^(1/(-1+(1+sigman)/(1+sigman/(1+sigmah)))));
empSCSS=((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(-1/(1+sigmah+sigman))*HSCSS^(1/(1+sigman/(1+sigmah)));
empSKSS=((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(-1/(1+sigmah+sigman))*HSKSS^(1/(1+sigman/(1+sigmah)));
HrSCSS=HSCSS/empSCSS;
HrSKSS=HSKSS/empSKSS;
UHSCSS=A_HC*((1+sigman)/(1+sigman/(1+sigmah)))*HSCSS^(-1+(1+sigman)/(1+sigman/(1+sigmah)));
UHSKSS=A_HK*((1+sigman)/(1+sigman/(1+sigmah)))*HSKSS^(-1+(1+sigman)/(1+sigman/(1+sigmah)));
unempSS=(empSCSS+empSKSS-(empCSS+empKSS))/(empSCSS+empSKSS);
QKSS=1;
QCDSS=1;
QCHSS=1;
UCSS=1;
UKSS=1;
XiBSS=1;
XiDSS=1;
XiHSS=1;
RL1SS=RSS;
RL2SS=RSS;
RL3SS=RSS;
RL4SS=RSS;
RL5SS=RSS;
RL6SS=RSS;
RL7SS=RSS;
DIFFREALECSS =exp( log(MUCSS));
DIFFREALEIKSS =exp( log(MUKSS));
DIFFREALECDSS =exp( log(MUKSS));
DIFFREALECHSS =exp( log(MUCSS));
DIFFREALWSS =exp( log(MUCSS) );
DIFFREALGDPSS =exp( (1-s_k)*log(MUCSS)+(s_k)*log(MUKSS));

%end_steady_state;

```

```

%trends;

DIFFREALGDPSS_obs=(1-s_k)*log(MUCSS)*100+(s_k)*log(MUKSS)*100;
DIFFREALECSS_obs=log(MUCSS)*100;
DIFFREALEIKSS_obs=log(MUKSS)*100;
DIFFREALECDSS_obs=log(MUKSS)*100;
DIFFREALECHSS_obs=log(MUCSS)*100;
DIFFREALWSS_obs=log(MUCSS)*100;
INFCNASS_obs=(1-s_ecdc)*log(INFCSS)*100+s_ecdc*log(INFKSS)*100;
INFCORSS_obs=(1-s_ecdc)*log(INFCSS)*100+s_ecdc*log(INFKSS)*100;
INFKSS_obs=log(INFCSS)*100-log(MUKSS)*100+log(MUCSS)*100;
RSS_obs=log(RSS)*100;
RT2SS_obs=log(RT2SS)*100;
unempSS_obs=100*log(unempSS);

%end_trends;

for i=1:NumberofParameters
    paramname=deblank(M_.param_names(i,:));
    eval(['M_.params(' int2str(i) ')=' paramname ';'']);
end;

ys = [
RCSS
RKSS
WCSS
WKSS
YCSS
YKSS
MCCSS
MCKSS
KCSS
KKSS
PKBSS
RSS
LSS
QKSS
HCSS
HSCSS
HKSS
HSKSS
UHCSS
UHSCSS

```

UHKSS
UHSKSS
empCSS
HrCSS
empKSS
HrKSS
empSCSS
HrSCSS
empSKSS
HrSKSS
unempSS
EIKSS
ECSS
INFWCSS
INFWKSS
INFCSS
INFKSS
ONE
ONE
DIFFREALGDPSS
DIFFREALECSS
DIFFREALEIKSS
DIFFREALWSS
AHSS
INFGDPSS
INFCNASS
INFCORSS
ONE
ONE
INFC10SS
ECDSS
KCDSS
RCDSS
QCDSS
KCHSS
RCHSS
ECHSS
QCHSS
KCDSS
KCHSS
USS
USS
DIFFREALECHSS
DIFFREALECDSS
beta_
ONE

```
ONE
ONE
MUZKSS
MUZMSS
ONE
MUCSS
MUKSS
ONE
ONE
ONE
RL1SS
RL2SS
RL3SS
RL4SS
RL5SS
RL6SS
RL7SS
RT2SS
DIFFREALGDPSS_obs
DIFFREALECSS_obs
DIFFREALEIKSS_obs
DIFFREALECDSS_obs
DIFFREALECHSS_obs
DIFFREALWSS_obs
ONE
INFCNASS_obs
INFCORSS_obs
INFKSS_obs
RSS_obs
RT2SS_obs
unempSS_obs
];
```

This code is written to file `srcedo/Dynare.edo.steadystate.m`.

A.3 linearized.mod

24 $\langle \text{srcedo/linearized.mod } 24 \rangle \equiv$

```
var RC RK WC WK YC YK MCC MCK KC KK PKB R L QK HC HSC HK HSK UHC UHSC UHK UHSK er
DIFFREALGDP_obs DIFFREALEC_obs DIFFREALEIK_obs DIFFREALECD_obs DIFFREALECH_obs DIFFRE
```

```
varexo eHG eXiL eLpref eR eMUZK eMUZM ePMKC ePMKK eEFFECH eEFFECD eEFFK eB eSTAR;
```

```
parameters
```

```
h r_inf r_y r_dy phi_pc phi_H phi_wc phi_ic phi_cd phi_ech gam_pc gam_wc gam_ic gam_
rho_EFFECD rho_HG rho_EFFECH tp2 ONE MUZMSS MUZKSS r_dinf rpr phi_u rho_MUZK rho_MU
theta_k theta_wc theta_wk g_y a_ks s_AS gam_h gam_ech s_k s_ecdc eta_cnn eta_cd eta
icoef mu_betarl MUZCSS RCSS RKSS WCSS WKSS YCSS YKSS MCCSS MCKSS KCSS KKSS LSS HCSS
MUCSS MUKSS AHSS ECDSS KCDSS QCDSS RCDSS ECHSS KCHSS QCHSS RCHSS UKSS UCSS USS MUKSS
INFCNASS INFCORSS INFC10SS RT2SS beta_0 beta_2 beta_PYSS AA DD RR
eta_cd_eta_cnn eta_ch_eta_cnn Rnr ycbi_ykb hc_hk HSS ycbi ykb YYSS s_k_ecd s_c_ech s
sig_HG sig_XiL sig_lpref sig_R sig_MUZK sig_MUZM sig_PMKC sig_PMKK sig_EFFECH sig_E
HSKSS HSCSS HrCSS HrKSS A_HC sigman sigmah A_HK xsi_NC xsi_HrC xsi_NK xsi_HrK rho_XiL
empCSS empKSS HrSKSS HrSCSS empSCSS empSKSS UHCSS UHKSS UHSCSS UHSKSS unempSS DIFFRE
DIFFREALECHSS DIFFREALEIKSS DIFFREALWSS RL1SS RL2SS RL3SS RL4SS RL5SS
RL6SS RL7SS DIFFREALGDPSS_obs DIFFREALECSS_obs DIFFREALEIKSS_obs DIFFREALECDSS_obs
DIFFREALECHSS_obs DIFFREALWSS_obs INFCNASS_obs INFCORSS_obs INFKSS_obs
RSS_obs RT2SS_obs unempSS_obs;
```

```
//estimated_params;
```

```
h = 0.715162417869797;
r_inf = 1.46344163969035;
r_y = 0.263123294207851;
phi_pc = 3.54471453295450;
phi_H = 3.22894079106560;
phi_wc = 5.49395755514723;
phi_ic = 0.253308786976374;
phi_cd = 0.470089385005009;
phi_ech = 9.13986886546163;
gam_pc = 0.314488926051065;
gam_wc = -0.230018833252054;
sigman = 39.4075260618789;
sigmah = 21.8859803402692;
rho_R = 0.833200065745674;
rho_XiL = 0.263567746111198;
rho_lpref = 0.979092048897712;
rho_B = 0.895267027146152;
```



```
rho_STAR      = 0.909187927454138;
rho_EFFK      = 0.937829274540004;
rho_EFFECD    = -0.240286975088701;
rho_HG        = 0.582395471123139;
rho_EFFECH    = 0.877235725078934;
tp2           = 0.000307314910763576;
sig_HG        = 0.579315931803017;
sig_XiL       = 2.49313873916751;
sig_lpref     = 5.66476748114241;
sig_R         = 0.124100461010359;
sig_MUZK      = 0.936167718269030;
sig_MUZM      = 0.597390920898135;
sig_PMKC      = 0.451830653200989;
sig_PMKK      = 0.685376191952156;
sig_EFFECH    = 0.514704527091087;
sig_EFFECD    = 9.11199585973990;
sig_EFFK      = 0.402779878811407;
sig_B         = 0.295232712196573;
sig_STAR      = 0.104877885500673;
//end_estimated_params;
```

```
//calibrated_params;
r_dy = 0;
ONE = 1;
MUZKSS = 1.009250;
MUZMSS = 1.001000;
gam_ic = 1.0;
gam_icd = 1.0;
r_dinf = 0;
rpr = 0.965;
phi_u = 1;
rho_MUZK = 0;
rho_MUZM = 0;
pbeta = 0.99862;
delta_ = 0.03;
h_cd = 0.0;
h_ch = 0.0;
delta_cd = 0.055;
delta_ch = 0.0035;
alpha_ = 0.26;
theta_c = 7;
theta_k = 7;
unempSS = .06;
g_y = 0.0;
a_ks = 0.2;
s_AS = 0.2;
```

```

gam_h = 1;
gam_ech = 1;
icoef = 3;
betarl = .958;
//end_calibrated_params;

//free_params;
//A_HC;
//A_HK;
//xsi_NC;
//xsi_HrC;
//xsi_NK;
//xsi_HrK;
//theta_wc;
//theta_wk;
//infkbar;
//infcbars;
//infwcbar;
//infwkbar;
//Pybar;
//Yybar;
//mu_yc;
//mu_yk;
//s_k;
//s_ecdc;
//eta_cnn;
//eta_cd;
//eta_ch;
//mu_;
//end_free_params;

//calibrated ME

//*****
//MODEL BLOCK
//*****

model;
(RCSS*exp(RC))-(MCCSS*exp(MCC))*(YCSSL*exp(YC))/(USS*exp(UC))/(KCSS*exp(KC(-1)))*alpha_
(RKSSL*exp(RK))-(MCKSSL*exp(MCK))*(YKSSL*exp(YK))/(USS*exp(UK))/(KKSSL*exp(KK(-1)))*alpha_
(WCSS*exp(WC))-(MCCSSL*exp(MCC))*(YCSSL*exp(YC))/(HCSSL*exp(HC))*(1-alpha_)=0;
(WKSSL*exp(WK))-(MCKSSL*exp(MCK))*(YKSSL*exp(YK))/(HKSSL*exp(HK))*(1-alpha_)=0;
(YCSSL*exp(YC))-(USS*exp(UC))*(KCSS*exp(KC(-1)))/(MUKSSL*exp(MUK))^alpha_*(HCSSL*exp(HC))
(YKSSL*exp(YK))-(USS*exp(UK))*(KKSSL*exp(KK(-1)))/(MUKSSL*exp(MUK))^alpha_*(HKSSL*exp(HK))
(MCCSSL*exp(MCC))*(YCSSL*exp(YC))*theta_c-(theta_c-1)*(YCSSL*exp(YC))-100*phi_pc*((INFCS

```

```

(MCKSS*exp(MCK))*(YKSS*exp(YK))*theta_k/(PKBSS*exp(PKB))-(theta_k-1)*(YKSS*exp(YK))-100*phi_pc*
(QKSS*exp(QK))-beta_*(1/(ONE*exp(EFFK)))*(((1-delta_)*(QKSS*exp(QK(+1)))+(RCSS*exp(RC(+1)))*(US
(QKSS*exp(QK))-beta_*(1/(ONE*exp(EFFK)))*(((1-delta_)*(QKSS*exp(QK(+1)))+(RKSS*exp(RK(+1)))*(US
(LSS*exp(L))-(beta_*exp(betas))*(RSS*exp(R))/rpr/(INFCSS*exp(INFC(+1)))/(MUCSS*exp(MUC(+1)))*(L
ln((RSS*exp(R))/RSS)-rho_R*ln((RSS*exp(R(-1)))/RSS)-(1-rho_R)*(r_inf*ln((INFCNASS*exp(INFCNA))/
(LSS*exp(L))-eta_cnn/(ECSS*exp(EC))-h*(ECSS*exp(EC(-1)))/(MUCSS*exp(MUC)))+eta_cnn*beta_*h/(M
(KKSS*exp(KK))-(1-delta_)*(KKSS*exp(KK(-1)))/(MUKSS*exp(MUK)))+(KCSS*exp(KC))-(1-delta_)*(KCSS*exp

// XXXXXXXXXXXXXXXXXXXXXXXXXXXX
// labor block
// TOTAL LABOR INPUT (called "(LSS*exp(L))" in the paper, I kept the "H" notation of the origin
-100+(UHCSS*exp(UHC))*theta_wc-(theta_wc-1)*(WCSS*exp(WC))-100*phi_wc*((INFWCSS*exp(INFWC))-gam
(UHSCSS*exp(UHSC))-(WCSS*exp(WC))+phi_H/10*((HSCSS*exp(HSC))/(HSKSS*exp(HSK))-gam_h*(HSCSS*exp
-100+(UHKSS*exp(UHK))*theta_wk-(theta_wk-1)*(WKSS*exp(WK))-100*phi_wc*((INFWKSS*exp(INFWK))-gam
(UHSKSS*exp(UHSK))-(WKSS*exp(WK))-phi_H/10*((HSCSS*exp(HSC))/(HSKSS*exp(HSK))-gam_h*(HSCSS*exp
(UHCSS*exp(UHC))*(LSS*exp(L))*(ONE*exp(Lpref))-A_HC*((1+sigman)/(1+sigman/(1+sigmah)))*((HSCSS*exp
(UHSCSS*exp(UHSC))*(LSS*exp(L))*(ONE*exp(Lpref))-A_HC*((1+sigman)/(1+sigman/(1+sigmah)))*((HSCSS*exp
(UHKSS*exp(UHK))*(LSS*exp(L))*(ONE*exp(Lpref))-A_HK*((1+sigman)/(1+sigman/(1+sigmah)))*((HSCSS*exp
(UHSKSS*exp(UHSK))*(LSS*exp(L))*(ONE*exp(Lpref))-A_HK*((1+sigman)/(1+sigman/(1+sigmah)))*((HSCSS*exp
(empCSS*exp(empC))-((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(-1/(1+sigmah+sigman))*(HSCSS*exp(HC))^(-1/
(HrCSS*exp(HrC))-((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(-1/(1+sigmah+sigman))*(HSCSS*exp(HC))^(-1/
(empKSS*exp(empK))-((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(-1/(1+sigmah+sigman))*(HSKSS*exp(HK))^(-1/
(HrKSS*exp(HrK))-((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(-1/(1+sigmah+sigman))*(HSKSS*exp(HK))^(-1/
(empSCSS*exp(empSC))-((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(-1/(1+sigmah+sigman))*(HSCSS*exp(HSC))^(-1/
(HrSCSS*exp(HrSC))-((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(-1/(1+sigmah+sigman))*(HSCSS*exp(HSC))^(-1/
(empSKSS*exp(empSK))-((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(-1/(1+sigmah+sigman))*(HSKSS*exp(HSK))^(-1/
(HrSKSS*exp(HrSK))-((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(-1/(1+sigmah+sigman))*(HSKSS*exp(HSK))^(-1/
(unempSS*exp(unemp))-((empSCSS*exp(empSC))+(empSKSS*exp(empSK))-((empCSS*exp(empC))+(empKSS*exp
(PKBSS*exp(PKB))-(1-100*phi_ic*(EIKSS*exp(EIK))-gam_ic*(EIKSS*exp(EIK(-1)))-(1-gam_ic)*EIKSS)/
(YCSS*exp(YC))-(ECSS*exp(EC))-(ECHSS*exp(ECH))-0.2*YCSS*(ONE*exp(HG))=0;
ln((INFWCSS*exp(INFWC))-ln((WCSS*exp(WC)))+ln((WCSS*exp(WC(-1)))-ln((MUCSS*exp(MUC)))-ln((INFW
ln((INFWKSS*exp(INFWK))-ln((WKSS*exp(WK)))+ln((WKSS*exp(WK(-1)))-ln((MUCSS*exp(MUC)))-ln((INFW
ln((INFKSS*exp(INFK))-ln((INFCSS*exp(INFC)))-ln((PKBSS*exp(PKB)))+ln((PKBSS*exp(PKB(-1)))+ln((
(YKSS*exp(YK))-(EIKSS*exp(EIK))-(ECDSS*exp(ECD))-0.2*YKSS*(ONE*exp(HG))=0;
ln((ONE*exp(DIFFNORMGDP))-(1-s_k)*(ln((YCSS*exp(YC))-ln((YCSS*exp(YC(-1)))))-s_k*(ln((YKSS*exp
ln((ONE*exp(NORMINFGDP)))-s_k*(ln((PKBSS*exp(PKB))-ln((PKBSS*exp(PKB(-1))))))=0;
ln((DIFFREALGDPSS*exp(DIFFREALGDP))-ln((ONE*exp(DIFFNORMGDP))-(1-s_k)*ln((MUCSS*exp(MUC)))-s_k*
ln((DIFFREALECSS*exp(DIFFREALEC))-ln((ECSS*exp(EC))+ln((ECSS*exp(EC(-1))))-ln((MUCSS*exp(MUC))
ln((DIFFREALEIKSS*exp(DIFFREALEIK))-ln((EIKSS*exp(EIK))+ln((EIKSS*exp(EIK(-1))))-ln((MUKSS*exp

// Identities
ln((DIFFREALWSS*exp(DIFFREALW)))-HCSS/AHSS*(ln((INFWCSS*exp(INFWC)))-HKSS/AHSS*(ln((INFWKSS*exp

// XXXXXXXXXXXXXXXXXXXXXXXX
// Aggregate hours equals agg hours in each sector

```

```

(AHSS*exp(AH))-(HCSS*exp(HC))-(HKSS*exp(HK))=0;
ln((INFGDPSS*exp(INFGDP)))-ln((INFCSS*exp(INFC)))-ln((YCSSL*exp(YC))*(MUCSS*exp(MUC)))-
ln((INFCNASS*exp(INFCNA)))-(1-s_ecdc)*ln((INFCSS*exp(INFC)))-s_ecdc*ln((INFKSS*exp(INFK)))-
ln((INFCORSS*exp(INFCOR)))-(1-s_ecdc)*ln((INFCSS*exp(INFC)))-s_ecdc*ln((INFKSS*exp(INFK)))-
ln((ONE*exp(GAP)))-(1-s_k)*ln((YCSSL*exp(YC))/YCSSL-s_k*ln((YKSS*exp(YK))/YKSS)=0;
ln((ONE*exp(PFGAP)))-(1-alpha_)*((1-s_k)*ln((HCSSL*exp(HC))/HCSSL+s_k*ln((HKSS*exp(HK))/HKSS))-
ln((INFC10SS*exp(INFC10)))-betarl*ln((INFC10SS*exp(INFC10(+1))))-(1-betarl)*ln((INFC10SS*exp(INFC10(+1))))=0;

// See Section 8: Data Identities

// new equations
// Durable Block

(KCDSSL*exp(KD))-(1-delta_cd)*(KCDSSL*exp(KD(-1)))/(MUKSSL*exp(MUK))-(ECDSSL*exp(ECD))=0;
(LSSL*exp(L))*(RCDSSL*exp(RCD))-eta_cd/((KCDSSL*exp(KD(-1)))/(MUKSSL*exp(MUK))-h_cd*(KCDSSL*exp(KD)))-
(QCDSSL*exp(QCD))-beta_*(1/(ONE*exp(EFFECD)))*(LSSL*exp(L(+1)))/(LSSL*exp(L))/(MUKSSL*exp(MUK))-
(PKBSSL*exp(PKB))-(QCDSSL*exp(QCD))*(1-100*phi_cd*((ECDSSL*exp(ECD))-gam_icd*(ECDSSL*exp(ECD))))=0;

// Housing Block
(LSSL*exp(L))*(RCHSSL*exp(RCH))-eta_ch/((KCHSSL*exp(KCH(-1)))/(MUCSSL*exp(MUC))-h_ch*(KCHSSL*exp(KCH)))-
(QCHSSL*exp(QCH))-beta_*(1/(ONE*exp(EFFECH)))*(LSSL*exp(L(+1)))/(LSSL*exp(L))/(MUCSSL*exp(MUC))-
1*(ECHSSL*exp(ECH))+(1-delta_ch)*(KCHSSL*exp(KCH(-1)))/(MUCSSL*exp(MUC))-(KCHSSL*exp(KCH))-
1-(QCHSSL*exp(QCH))*(1-100*phi_ech*((ECHSSL*exp(ECH))-gam_ech*(ECHSSL*exp(ECH(-1))))-(1-gam_ech*(ECHSSL*exp(ECH(-1)))))-
ln((KCDSSL*exp(KD(-1))))-ln((KCDSSL*exp(LAGKD)))=0;
ln((KCHSSL*exp(KCH(-1))))-ln((KCHSSL*exp(LAGKCH)))=0;
(RKSSL*exp(RK))-(QKSSL*exp(QK))*mu_*(USS*exp(UK))^(1/phi_u)=0;
(RCSSL*exp(RC))-(QKSSL*exp(QK))*mu_*(USS*exp(UC))^(1/phi_u)=0;
ln((DIFFREALECHSSL*exp(DIFFREALECH)))-ln((MUCSSL*exp(MUC)))-ln((ECHSSL*exp(ECH)))+ln((ECHSSL*exp(ECH(-1))))-
ln((DIFFREALECDSSL*exp(DIFFREALECD)))-ln((MUKSSL*exp(MUK)))-ln((ECDSSL*exp(ECD)))+ln((ECHSSL*exp(ECH(-1))))-
ln((beta_*exp(betas))/beta_)-rho_B*ln((beta_*exp(betas(-1))))/beta_-eB=0;
ln((ONE*exp(XiL)))-rho_XiL*ln((ONE*exp(XiL(-1))))-eXiL=0;
ln((ONE*exp(Lpref)))-rho_lpref*ln((ONE*exp(Lpref(-1))))-eLpref=0;
ln((ONE*exp(EFFK)))-rho_EFFK*ln((ONE*exp(EFFK(-1))))-eEFFK=0;
ln((MUZKSSL*exp(MUZK))/MUZKSSL)-eMUZK=0;
ln((MUZMSSL*exp(MUZM))/MUZMSSL)-eMUZM=0;
ln((ONE*exp(HG)))-rho_HG*ln((ONE*exp(HG(-1))))-eHG=0;
ln((MUCSSL*exp(MUC)))-ln((MUZMSSL*exp(MUZM)))-alpha_*ln((MUZKSSL*exp(MUZK)))=0;
ln((MUKSSL*exp(MUK)))-ln((MUZMSSL*exp(MUZM)))-ln((MUZKSSL*exp(MUZK)))=0;
ln((ONE*exp(EFFECD)))-rho_EFFECD*ln((ONE*exp(EFFECD(-1))))-eEFFECD=0;
ln((ONE*exp(EFFECH)))-rho_EFFECH*ln((ONE*exp(EFFECH(-1))))-eEFFECH=0;
ln((ONE*exp(STAR)))-rho_STAR*ln((ONE*exp(STAR(-1))))-eSTAR=0;
ln((RL1SSL*exp(RL1)))-ln((RSS*exp(R(+1))))=0;
ln((RL2SSL*exp(RL2)))-ln((RL1SSL*exp(RL1(+1))))=0;
ln((RL3SSL*exp(RL3)))-ln((RL2SSL*exp(RL2(+1))))=0;
ln((RL4SSL*exp(RL4)))-ln((RL3SSL*exp(RL3(+1))))=0;
ln((RL5SSL*exp(RL5)))-ln((RL4SSL*exp(RL4(+1))))=0;

```

```

ln((RL6SS*exp(RL6))) - ln((RL5SS*exp(RL5(+1))))=0;
ln((RL7SS*exp(RL7))) - ln((RL6SS*exp(RL6(+1))))=0;
ln((RT2SS*exp(RT2))) - tp2 - 0.125*(ln((RSS*exp(R)))) + ln((RL1SS*exp(RL1))) + ln((RL2SS*exp(RL2)))

//measurement_equations;
DIFFREALGDP_obs = DIFFREALGDP + DIFFREALGDPSS_obs;
DIFFREALEC_obs = DIFFREALEC + DIFFREALECSS_obs;
DIFFREALEIK_obs = DIFFREALEIK + DIFFREALEIKSS_obs;
DIFFREALECD_obs = DIFFREALECD + DIFFREALECDSS_obs;
DIFFREALECH_obs = DIFFREALECH + DIFFREALECHSS_obs;
DIFFREALW_obs = DIFFREALW + DIFFREALWSS_obs;
AH_obs = AH;
INFCNA_obs = INFCNA + INFCNASS_obs;
INFCOR_obs = INFCOR + INFCORSS_obs;
INFK_obs = INFK + INFKSS_obs;
R_obs = R + RSS_obs;
RT2_obs = RT2 + RT2SS_obs;
unemp_obs = unemp + unempSS_obs;

//end_measurement_equations;

end;

varobs DIFFREALGDP_obs DIFFREALEC_obs DIFFREALEIK_obs DIFFREALECD_obs DIFFREALECH_obs DIFFREALW_obs

shocks;
var eHG;
stderr sig_HG;
var eXiL;
stderr sig_XiL;
var eLpref;
stderr sig_lpref;
var eR;
stderr sig_R;
var eMUZK;
stderr sig_MUZK;
var eMUZM;
stderr sig_MUZM;
var ePMKC;
stderr sig_PMKC;
var ePMKK;
stderr sig_PMKK;
var eEFFECH;
stderr sig_EFFECH;
var eEFFECD;
stderr sig_EFFECD;

```

```

var eEFFK;
stderr sig_EFFK;
var eB;
stderr sig_B;
var eSTAR;
stderr sig_STAR;

```

```

var DIFFREALGDP_obs;
stderr 0.3;
var DIFFFREALEC_obs;
stderr 0.1;
var DIFFFREALEIK_obs;
stderr 1.5;
var DIFFFREALECD_obs;
stderr 1.5;
var DIFFFREALECH_obs;
stderr 1.5;
var DIFFREALW_obs;
stderr 0.3;
var AH_obs;
stderr 0.3;
var INFCNA_obs;
stderr 0.5;
var INFCOR_obs;
stderr 0.05;
var INFK_obs;
stderr 0.2;
var RT2_obs;
stderr 0.1;
var unemp_obs;
stderr 4;
end;

```

```

steady;

```

```

estimated_params;

```

h	, .673	, -1	, 1	, uniform_pdf	,,, -1
r_inf	, 1.461	, -999	, 999	, normal_pdf	, 1.5000
r_y	, 0.214	, -999	, 999	, normal_pdf	, 0.125
phi_pc	, 3.126	, 0	, 999	, gamma_pdf	, 4.0000
phi_H	, 4.064	, 0	, 999	, gamma_pdf	, 4.0000
phi_wc	, 5.119	, 0	, 999	, gamma_pdf	, 4.0000
phi_ic	, .325	, 0	, 999	, gamma_pdf	, 4.0000
phi_cd	, .651	, 0	, 999	, gamma_pdf	, 4.0000
phi_ech	, 10.948	, 0	, 999	, gamma_pdf	, 4.0000

```

gam_pc      , 0.386      , -999      , 999      , normal_pdf      , 0.000      , 0.250
gam_wc      , 0.213      , -999      , 999      , normal_pdf      , 0.000      , 0.250
sigman      , 1.25       , 0        , 999      , gamma_pdf      , 1.25       , 12.5
sigmah      , 10         , 0        , 999      , gamma_pdf      , 10         , 100
rho_R       , 0.654       , -1       , 1        , normal_pdf      , 0.5        , 0.25
rho_XiL     , 0.654       , -1       , 1        , normal_pdf      , 0.5        , 0.25
rho_lpref   , 0.954       , -1       , 1        , normal_pdf      , 0.5        , 0.25
rho_B       , 0.825       , -1       , 1        , normal_pdf      , 0          , 0.5
rho_STAR    , 0.825       , -1       , 1        , normal_pdf      , 0          , 0.5
rho_EFFK    , 0.850       , -1       , 1        , normal_pdf      , 0          , 0.5
rho_EFFECD  , .230        , -1       , 1        , normal_pdf      , 0          , 0.5
rho_HG      , 0.596       , 0        , 1        , beta_pdf       , 0.5        , 0.015
rho_EFFECH  , 0.844       , -1       , 1        , normal_pdf      , 0          , 0.5
tp2         , 0.001       , -999      , 999      , normal_pdf      , 0.0        , 0.000

```

```

stderr eHG      , .745      , 0.0001   , 999      , inv_gamma_pdf  , 1.772454   , Inf
stderr eXiL     , 3.621     , 0.0001   , 999      , inv_gamma_pdf  , 1.772454   , Inf
stderr eLpref   , 1.621     , 0.0001   , 999      , inv_gamma_pdf  , 1.772454   , Inf
stderr eR       , 0.165     , 0.0001   , 999      , inv_gamma_pdf  , 0.354491   , Inf
stderr eMUZK    , .834      , 0.0001   , 999      , inv_gamma_pdf  , 0.443113   , Inf
stderr eMUZM    , .484      , 0.0001   , 999      , inv_gamma_pdf  , 0.443113   , Inf
stderr ePMKC    , .391      , 0.0001   , 999      , inv_gamma_pdf  , 0.354491   , Inf
stderr ePMKK    , .552      , 0.0001   , 999      , inv_gamma_pdf  , 0.354491   , Inf
stderr eEFFECH  , .526      , 0.0001   , 999      , inv_gamma_pdf  , 1.772454   , Inf
stderr eEFFECD  , 13.349    , 0.0001   , 999      , inv_gamma_pdf  , 1.772454   , Inf
stderr eEFFK    , .499      , 0.0001   , 999      , inv_gamma_pdf  , 1.772454   , Inf
stderr eB       , 0.5       , 0.0001   , 999      , inv_gamma_pdf  , 1.772454   , Inf
stderr eSTAR    , 0.05      , 0.0001   , 999      , inv_gamma_pdf  , 0.354491   , Inf
end;

```

```

options_.order = 1;
options_.jacobian_flag = 1;
options_.nonlin = 1;

```

```

stoch_simul(order=1,irf=40,nograph);

```

This code is written to file `srcedo/linearized.mod`.

A.4 linearized_steadystate.m

```

32  <srcdo/linearized.steadystate.m 32>≡
    function [ys,check] = linearized_steadystate(ys,exe)
        global M_

    check = 0;

    NumberofParameters=M_.param_nbr;
    for i=1:NumberofParameters
        paramname=deblank(M_.param_names(i,:));
        eval([paramname ' =M_.params(' int2str(i) ');']);
    end;

    %start_steady_state;

    beta_0 = pbeta;
    beta_2 = pbeta*rpr; % s.s. funds rate premium
    beta_ = beta_2;
    MUZCSS=1;
    ONE=1;
    USS=1;
    MUKSS=MUKSS*MUZMSS;
    MUCSS=MUKSS^alpha_*MUZMSS;
    MUKSShabit=MUKSS;
    MUCSShabit=MUCSS;
    PKBSS=theta_k/(theta_k-1)*(theta_c-1)/theta_c;
    PYSS=1;
    MCCSS=(theta_c-1)/theta_c;
    MCKSS=(theta_k-1)/theta_k;
    RKSS=MUKSS/beta_2-(1-delta_);
    RCSS=MUKSS/beta_2-(1-delta_);
    RCHSS=MUCSS/beta_2-(1-delta_ch); % Housing sector
    RCDSS=MUKSS/beta_2-(1-delta_cd); % Durable sector
    USS=1;
    mu_=RCSS;
    AA=alpha_/RKSS*MCKSS;
    DD = 0.135;
    RR = 0.075;
    eta_cnn=1;
    eta_cd_eta_cnn=DD/((MUKSShabit-beta_2*h_cd)/(1-beta_2*h/MUCSShabit)*(1-h/MUCSShabit));
    eta_ch_eta_cnn=RR/((MUCSShabit-beta_2*h_ch)/(1-beta_2*h/MUCSShabit)*(1-h/MUCSShabit));
    eta_ch=eta_ch_eta_cnn;
    eta_cd=eta_cd_eta_cnn;
    DD=eta_cd_eta_cnn*(MUKSShabit-beta_2*h_cd)/(1-beta_2*h/MUCSShabit)*(1-h/MUCSShabit);
    RR=eta_ch_eta_cnn*(MUCSShabit-beta_2*h_ch)/(1-beta_2*h/MUCSShabit)*(1-h/MUCSShabit);

```



```

Rnr=(1-(1-delta_)/MUKSS)*AA*MUKSS;
ycbi_ykb=((1-s_AS)-Rnr)/((DD*(1-s_AS)/(1+RR))+Rnr);
hc_hk=ycbi_ykb*(RCSS*MCKSS/(RKSS*MCCSS))^(alpha_/(1-alpha_));
HSS=0.25;
AHSS=HSS;
HKSS=HSS/(1+hc_hk);
HCSS=HSS-HKSS;
HrCSS=1/3;
HrKSS=1/3;
empCSS=HCSS/HrCSS;
empKSS=HKSS/HrKSS;
ycbi=HCSS*(AA)^(alpha_/(1-alpha_));
ykb=HKSS*(AA)^(alpha_/(1-alpha_));
YCSS=ycbi;
YKSS=ykb;
KCSS=AA*ycbi*MUKSS;
KKSS=AA*ykb*MUKSS;
ECHSS=RR/(1+RR)*ycbi*(1-s_AS);
ECSS=1/(1+RR)*ycbi*(1-s_AS);
ECDSS=DD*PKBSS*ECSS;
EIKSS=(1-(1-delta_)/MUKSS)*(KCSS+KKSS);
KCDSS=ECDSS/(1-(1-delta_cd)/MUKSS);
KCHSS=ECHSS/(1-(1-delta_ch)/MUCSS);
YYSS=(YCSS+YKSS*PKBSS)/PYSS;
s_k_ecd=ECDSS/YKSS;
s_c_ech=ECHSS/YCSS;
s_k_eik=EIKSS/YKSS;
s_yc = (YCSS/YYSS);
s_ecdc=PKBSS*ECDSS/(ECSS+PKBSS*ECDSS+(MUCSS/beta_2-1+delta_ch)*KCHSS);
INFCNASS=exp(.02/4);
INFCSS = INFCNASS*((MUZCSS/MUZKSS)^(1-alpha_))^(s_ecdc);
INFCORSS=INFCNASS;
INFKSS=INFCSS*(MUZCSS/MUZKSS)^(1-alpha_);
INFWCSS=INFCSS*MUZKSS^alpha_*MUZMSS;
INFWKSS=INFWCSS;
RSS=INFCSS/beta_0*MUCSS;
RT2SS=exp(tp2)*RSS;
INFC10SS = INFCNASS;
IMPHSSS = RCHSS*KCHSS;
s_k=PKBSS*YKSS/YYSS;
INFGDPSS=INFCSS*(YCSS/YYSS)*INFKSS*(YKSS*PKBSS/(YYSS));
LSS=eta_cnn/(ECSS*(1-h/MUCSShabit))-eta_cnn*beta_2*h/(ECSS*(MUCSShabit-h));
WCSS=MCCSS*(1-alpha_)*YCSS/HCSS;
WKSS=MCKSS*(1-alpha_)*YKSS/HKSS;
xsiN_xsiH_C = ((HrCSS/empCSS)^(1+sigmah))/(1+1/sigmah);
xsiN_xsiH_K = ((HrKSS/empKSS)^(1+sigmah))/(1+1/sigmah);

```

```

gC = (1/(1+sigman) + 1/sigmah)*(xsiN_xsiH_C*(1+sigmah)/sigmah)^(-(1+sigman)/(1+sigman));
markup_xsiN_C = (HCSS^((1+sigmah)*(1+sigman)/(1+sigmah+sigman)-1))*gC/(LSS*WCSS);
gK = (1/(1+sigman) + 1/sigmah)*(xsiN_xsiH_K*(1+sigmah)/sigmah)^(-(1+sigman)/(1+sigman));
markup_xsiN_K = (HKSS^((1+sigmah)*(1+sigman)/(1+sigmah+sigman)-1))*gK/(LSS*WKSS);
markup_w = (1-unempSS)^((1+sigmah+sigman)/(1+sigmah) - 1 - sigman);
theta_wc = markup_w/(markup_w - 1); theta_wk = theta_wc;
A_HC=LSS*(theta_wc-1)/theta_wc*WCSS/(((1+sigman)/(1+sigman)/(1+sigmah)))*HCSS^(-1+(1+sigman));
A_HK=LSS*(theta_wk-1)/theta_wk*WKSS/(((1+sigman)/(1+sigman)/(1+sigmah)))*HKSS^(-1+(1+sigman));
xsi_NC=A_HC/((1/(1+sigman)+1/sigmah)*(HCSS^sigman/HrCSS^(1+sigman+sigmah))^(1+sigman));
xsi_NK=A_HK/((1/(1+sigman)+1/sigmah)*(HKSS^sigman/HrKSS^(1+sigman+sigmah))^(1+sigman));
xsi_HrC=xsi_NC*(1+sigmah)/sigmah*(HCSS^sigman/HrCSS^(1+sigman+sigmah));
xsi_HrK=xsi_NK*(1+sigmah)/sigmah*(HKSS^sigman/HrKSS^(1+sigman+sigmah));
UHCSS=A_HC*((1+sigman)/(1+sigman/(1+sigmah)))*HCSS^(-1+(1+sigman)/(1+sigman/(1+sigmah)));
UHKSS=A_HK*((1+sigman)/(1+sigman/(1+sigmah)))*HKSS^(-1+(1+sigman)/(1+sigman/(1+sigmah)));
HSCSS=(WCSS*LSS/(A_HC*((1+sigman)/(1+sigman/(1+sigmah))))^(1/(-1+(1+sigman)/(1+sigman/(1+sigmah)))));
HSKSS=(WKSS*LSS/(A_HK*((1+sigman)/(1+sigman/(1+sigmah))))^(1/(-1+(1+sigman)/(1+sigman/(1+sigmah)))));
empSCSS=((1+sigmah)/sigmah*xsi_NC/xsi_HrC)^(-1/(1+sigmah+sigman))*HSCSS^(1/(1+sigman));
empSKSS=((1+sigmah)/sigmah*xsi_NK/xsi_HrK)^(-1/(1+sigmah+sigman))*HSKSS^(1/(1+sigman));
HrSCSS=HSCSS/empSCSS;
HrSKSS=HSKSS/empSKSS;
UHSCSS=A_HC*((1+sigman)/(1+sigman/(1+sigmah)))*HSCSS^(-1+(1+sigman)/(1+sigman/(1+sigmah)));
UHSKSS=A_HK*((1+sigman)/(1+sigman/(1+sigmah)))*HSKSS^(-1+(1+sigman)/(1+sigman/(1+sigmah)));
unempSS=(empSCSS+empSKSS-(empCSS+empKSS))/(empSCSS+empSKSS);
QKSS=1;
QCDSS=1;
QCHSS=1;
UCSS=1;
UKSS=1;
XiBSS=1;
XiDSS=1;
XiHSS=1;
RL1SS=RSS;
RL2SS=RSS;
RL3SS=RSS;
RL4SS=RSS;
RL5SS=RSS;
RL6SS=RSS;
RL7SS=RSS;
DIFFREALECSS =exp( log(MUCSS));
DIFFREALEIKSS =exp( log(MUKSS));
DIFFREALECDSS =exp( log(MUKSS));
DIFFREALECHSS =exp( log(MUCSS));
DIFFREALWSS =exp( log(MUCSS) );
DIFFREALGDPSS =exp( (1-s_k)*log(MUCSS)+(s_k)*log(MUKSS));

%end_steady_state;

```


[illegible]

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
DIFFREALGDPSS_obs
DIFFREALECSS_obs
DIFFREALEIKSS_obs
DIFFREALECDSS_obs
DIFFREALECHSS_obs
DIFFREALWSS_obs
0
INFCNASS_obs
INFCORSS_obs
INFKSS_obs
RSS_obs
RT2SS_obs
unempSS_obs
];
```

This code is written to file `srcedo/linearized.steadystate.m`.

A.5 readme.txt

38 *<srcedo/readme.txt 38>*≡

How to run the model:

=====

In Matlab/Octave:

- 1) Download Dynare Version 4 from the Dynare website: <http://www.dynare.org/>
- 2) Download the EDO files in a folder you choose.
- 3) Start Matlab/Octave and change the current directory to the folder in step 2.
- 4) Link in Matlab/Octave the Dynare folder in the menu under file/Set Path (or use the command "addpath path/to/dynare").
- 5) Run the command "dynare linearized" or "dynare Dynare_edo" from the Matlab/Octave command window.

Content of the EDO folder:

=====

Dynare_edo.mod: Dynare model file containing the latest estimated parameters and nonlinear equations.

Dynare_edo_steadystate.mod: Dynare steady-state file computes the steady state of the nonlinear model.

linearized.mod: Dynare model file containing the latest estimated parameters and nonlinear equations.

linearized_steadystate.mod: Dynare steady-state file computes the steady state of the linearized model.

readme.txt: The file you are currently reading.

This code is written to file *srcedo/readme.txt*.

Appendix B

Notes, Bibliography and Indexes

B.1 Chunks

<srcedo/Dynare.edo.mod 9>
<srcedo/Dynare.edo.steadystate.m 18>
<srcedo/linearized.mod 24>
<srcedo/linearized.steadystate.m 32>
<srcedo/readme.txt 38>

B.2 Index