```
!variables:transition

    LGDP, LGDP_BAR, Y, G, GROWTH, GROWTH4, GROWTH_BAR, GROWTH4_BAR
    PIELTE, PIE4
    UNR, UNR_BAR, UNR_G, UNR_GAP
    CAPU, CAPU_BAR, CU_G, CAPU_GAP

!shocks:transition

    RES_LGDP_BAR, RES_G, RES_Y
    RES_PIELTE, RES_PIE
    RES_UNR_BAR, RES_UNR_G, RES_UNR_GAP
    RES_CAPU_BAR, RES_CU_G, RES_CAPU_GAP

!parameters

    lab_share, tau, growth_ss, rho1, rho2
    beta, omega
    parhist, lambda, unr_ss, alpha, phi1, phi2
    delta, kappa1, kappa2

!equations:transition

    % Output.
    LGDP = LGDP_BAR + Y;
    LGDP_BAR = LGDP_BAR{-1} + G/4 + RES_LGDP_BAR
        - lab_share*(UNR_BAR - UNR_BAR{-1}) -(1-lab_share)*(UNR_BAR{-1} - UNR_BAR{-20})/19;
    G = (1-tau)*G{-1} + tau*growth_ss + RES_G;
    Y = rho1*Y{-1} - rho2/100*(PIE4{-1} - PIELTE{-1}) + RES_Y;
    GROWTH = 4*(LGDP - LGDP{-1});
    GROWTH4 = LGDP - LGDP{-4};
    GROWTH_BAR = 4*(LGDP_BAR - LGDP_BAR{-1});
    GROWTH4_BAR = LGDP_BAR - LGDP_BAR{-4};

    % Inflation.
    PIELTE = PIELTE{-1} + RES_PIELTE;
    PIE4 = PIE4{-1} + beta*Y + omega*(Y - Y{-1}) + RES_PIE;

    % Unemployment.
    UNR_GAP = UNR_BAR - UNR;
    UNR_BAR = UNR_BAR{-1} + UNR_G
        - parhist/100*Y{-1} - lambda/100*(UNR_BAR{-1} - unr_ss) + RES_UNR_BAR;
    UNR_G = (1-alpha)*UNR_G{-1} + RES_UNR_G;
    UNR_GAP = phi1*UNR_GAP{-1} + phi2*Y + RES_UNR_GAP;

    % Capacity utilization.
    CAPU = CAPU_BAR + CAPU_GAP;
    CAPU_BAR = CAPU_BAR{-1} + CU_G + RES_CAPU_BAR;
    CU_G = (1 - delta)*CU_G{-1} + RES_CU_G;
```

```
    CAPU_GAP = kappa1*CAPU_GAP{-1} + kappa2*Y + RES_CAPU_GAP;

!variables:measurement

    GROWTH_
    PIE4_
    PIE4_PIELTE_
    D_CAPU_
    UNR_

    % These measurement variables are used to impose priors on unobservables.
    GROWTH_BAR_

!equations:measurement

    GROWTH_ = GROWTH;
    PIE4_ = PIE4;
    PIE4_PIELTE_ = PIE4 - PIELTE;
    D_CAPU_ = CAPU - CAPU{-1};
    UNR_ = UNR;

    % These measurement equations are used to impose priors on unobservbles.
    GROWTH_BAR_ = GROWTH_BAR;
```

# Set important dates and choose country

```
% |endhist| is the last period used for the estimation of the model
% parameters
endhist = qq(2010,3);

% |endproj| is the date until which the Kalman smoother is run
endproj = qq(2017,4);

% Type the two-letter country code of the country you would like to
% analyze. Make sure that file named data_|country|.csv containing data
% for that country exists in |InputData| subdirectory.
country = 'US';

% Choose a name for the version of the results
%version = 'VER0';
version = 'VER2';
```

# readModel.m – Read and parametrize the model.

This m-file reads and parses the model code file `ModYUK.model`, and creates a model object in Matlab memory. The model object is then available for us to run the various exercises.

## Contents


- Clearing the workspace
- First-pass parameters
- Read the model code, assign parameters and create a model object
- Alternative ways of assigning parameters
- Solve the model, and compute steady-state.
- Save the model object for future use.


## Clearing the workspace

Clearing the workspace at the beginning of each new task is a really good idea.

```
clear all;
close all;
home();
```

## First-pass parameters

We create a parameter database (note we use the word 'database' for a Matlab 'struct') and use it then to assign the model's parameters. Alternatively, you could first read the model, and only then create a parameter database, and assign the parameters. Whichever suits you better.

There is a parameter controlling the std deviation of each shock, called `'std_'` and the name of the shock. You needn't declare these in the model code file, they're created automatically, but you can treat them exactly the same way as the other parameters.

Finally, note that the values of parameters set here are immaterial for those parameters that are later estimated in `estimateParams.m`

```
% A new empty database (struct).
P = struct();

% Steady-state parameters.
P.unr_ss = 5.45;
P.growth_ss = 2.5;
P.lab_share = 0.7;

% Transitory parameters.
P.alpha = 0.4630;
P.beta = 0.1485;
P.omega = 0.1547;
```

```
P.rho1 = 0.8687;
P.kappa1 = 0.4056;
P.phi1 = 0.6298;
P.phi2 = 0.2628;
P.tau = 0.2208;
P.delta = 0.4970;
P.kappa2 = 1.2899;
P.parhist = 3.37;
P.rho2 = 5.67;
P.lambda = 5;

% Standard deviations of shocks
P.std_RES_LGDP_BAR = 1;
P.std_RES_Y = 0.7427;
P.std_RES_G = 1.1305;
P.std_RES_UNR_GAP = 0.1333;
P.std_RES_UNR_BAR = 0.0594;
P.std_RES_UNR_G = 0.0903;
P.std_RES_CAPU_GAP = 0.4137;
P.std_RES_CAPU_BAR = 0.2490;
P.std_RES_CU_G = 0.0715;
P.std_RES_PIE = 0.5349;
P.std_RES_PIELTE = 0.0955;
```

## Read the model code, assign parameters and create a model object

The `model` function reads the model code file specified as the first input argument, and creates a model object `m` in Matlab memory.

- The 'linear'(=true) option indicates that this is a linear model (and some more efficient algorithms can be used).
- The 'assign'(=P) option allows us to assign the model's parameters straight away when creating the model object. Alternatively, the parameters may be assigned after the model object has been created.

```
m = model(...
    'ModYUK.model',...
    'linear',true,...
    'assign',P);
display(m);
```

```
m =

linear model object: 1 parameterisation(s)
solution(s) available for no parameterisation(s)
user data: empty
```

## Alternative ways of assigning parameters

What follows is alternative syntax for assigning or changing parameters. Remember to re-solve the model, and re-compute the steady state whenever you change some of the parameters (not needed, though, when you only change std deviations). Note that every time you use the `estimate` function the model is re-solved. Therefore, it is not imperative that we solve the model below. It is, however, good practice and we do it.

```
m = assign(m,P);
m = assign(m,'lambda',5,'tau',0.2208);
m.lambda = 5;
m.tau = 0.2208;
```

## Solve the model, and compute steady-state.

```
m = solve(m);
m = sstate(m);
display(m);
```

```
m =

linear model object: 1 parameterisation(s)
solution(s) available for a total of 1 parameterisation(s)
user data: empty
```

## Save the model object for future use.

Saved using the IRIS `savestruct` function.

```
savestruct('model.mat',m);
```

# readData.m – Read data from a CSV file.

## Contents


- Clear the workspace
- Define dates and the country of interest.
- Download data from CSV file and load model
- Create measurement variables
- Check if any measurement variables are missing from the database
- Save data structure for future use.


## Clear the workspace

```
clear('all');
close('all');
home();
```

## Define dates and the country of interest.

```
readDefinitions;
```

## Download data from CSV file and load model

```
d = dbload(sprintf('InputData/data_%s.csv',country));
m = loadstruct('model.mat');
```

## Create measurement variables

```
d.GROWTH_ = 4*diff(d.LGDP);
d.PIE4_ = d.PIE4;
d.D_CAPU_ = diff(d.CAPU);
d.UNR_ = d.UNR;
d.PIE4_PIELTE_ = d.PIE4 - d.PIELTE;
```

## Check if any measurement variables are missing from the database

```
ylist = get(m,'ylist');
list = fieldnames(d);
missing = ylist - list;
if isempty(missing)
   disp('All measurement variables available.');
```

```
else
    disp('Measurement variables missing from the database:');
    disp(missing);
end
```

```
Measurement variables missing from the database:
    'GROWTH_BAR_'
```

## Save data structure for future use.

```
savestruct('data.mat',d);
```

# estimateParams.m – Estimate parameters using maximum regularised likelihood.

## Contents


- Clearing the workspace
- Load data and model
- Read important dates
- Display some summary information
- Adding priors on unobservables
- Some housekeeping
- Prior database.
- Estimate model.
- Run Kalman smoother and projections.
- Correct unit-root variables for an implicit constant.
- Save results in a country- and version- specific .mat file.


## Clearing the workspace

```
clear all;
close all;
home();
```

## Load data and model

```
d = loadstruct('data.mat');
m = loadstruct('model.mat');
```

## Read important dates

```
readDefinitions;
```

## Display some summary information

```
fprintf('%s...\n',country);
display(version);
starthist = get(d.GROWTH_,'start');
fprintf('start date = %s\n',dat2char(starthist));
fprintf('unr_ss = %g, growth_ss = %g, lab_share = %g\n',m.unr_ss,m.growth_ss, m.lab_share);
```

```
US...

version =

VER2

start date = 1967Q2
unr_ss = 5.45, growth_ss = 2.5, lab_share = 0.7
```

## Adding priors on unobservables

```
d.GROWTH_BAR_ = tseries(qq(2008,4):qq(2009,3),0.5);
```

## Some housekeeping

```
if ~(exist(sprintf('Results/Version%s',version),'dir') == 7)
    mkdir(sprintf('Results/Version%s',version))
end

%Write a short description of current version to a text file
fid = fopen(sprintf('Results/Version%s/Description%s.txt',version,version),'w');
fprintf(fid,'Adding a hard prior on GROWTH_BAR over 2008Q4:2009Q3. \r\n'); %Notepad doesn't recognize \n as new line
fclose(fid);
```

## Prior database.

The prior for each parameter is a truncated normal distribution. Each parameter is assigned four numbers:

[ prior mode, lower bound, upper bound, 1/(prior std dev before penalty)^2 ]

```
E = struct();

% Priors on model parameters
E.alpha   = [0.9, 0.1,  1,   1/0.05^2 ];
E.beta    = [0.4, 0.1,  Inf, 1/0.1^2 ];
E.omega   = [0.5, 0,    5,   1/0.1^2 ];
E.rho1    = [0.8, 0.1,  1,   1/0.05^2 ];
E.kappa1  = [0.1, 0.01, 1,   1/0.2^2 ];
E.phi1    = [0.8, 0,    1,   1/0.05^2 ];
E.phi2    = [0.3, 0,    1,   1/0.05^2 ];
E.tau     = [0.1, 0,    1,   1/0.05^2 ];
E.delta   = [0.5, 0,    1,   1/0.05^2 ];
E.kappa2  = [1.5, 0,    3,   1/0.5^2 ];
```

```
E.parhist = [3,   0,    100, 1/0.5^2 ];
E.rho2    = [5,   0,    100, 1/1^2 ];
E.lambda  = [2,   0.01, 100, 1/1^2 ];


% Priors on standard deviations of transition shocks
E.std_RES_Y        = [1,     0.01, 3, 1/0.1^2 ];
E.std_RES_G        = [1,     0.01, 3, 1/0.1^2 ];
E.std_RES_UNR_GAP  = [0.5,   0.01, 3, 1/0.1^2 ];
E.std_RES_UNR_BAR  = [0.1, 0.01, 3, 1/0.05^2 ];
E.std_RES_UNR_G    = [0.1, 0.01, 3, 1/0.05^2 ];
E.std_RES_CAPU_GAP = [0.4,   0.01, 3, 1/0.1^2 ];
E.std_RES_CAPU_BAR = [0.25,  0.01, 3, 1/0.05^2 ];
E.std_RES_CU_G     = [0.075, 0.01, 3, 1/0.01^2 ];
E.std_RES_PIE      = [0.5,   0.01, 3, 1/0.1^2 ];
E.std_RES_PIELTE   = [0.3,   0.01, 3, 1/0.1^2 ];
E.std_RES_LGDP_BAR  = [1/4,   0.01, 3, 1/(0.1/4)^2 ];
```

## Estimate model.

The call to the `estimate` function below uses the so-called maximum regularized likelihood estimation method (Ljung, 1999 – see presentation) to calculate estimates of the model parameters.

The output arguments of the `estimate` function are

- `Estar`, database with point estimates of model parameters
- `obj`, value of objective function (minus log-likelihood or weighted sum of prediction errors)
- `Grad`, Gradient at optimum,
- `Hess`, Hessian of log-posterior, and Hessian of log-prior (or penalty function) at optimum,
- `mstar`, a model object with parameter estimates updated

The input arguments of the `estimate` function are

- `m`, a model object to be estimated
- `d`, a database with measurement variables
- `starthist:endhist`, date range for estimation
- `E`, prior database
- `penalty`, a scalar that governs the overall tightness of the priors on the parameters (0 = maximum likelihood)
- `noSolution`, tells the optimization routine what to do when it finds anything other than a unique solution. `Penalty` returns a large value of the objective function.

```
p = 0.1;
[Estar,obj,Grad,Hess,mstar] =...
    estimate(m,d,starthist:endhist,E,...
    'penalty',p,'tolx',1e-8,'tolfun',1e-8,'noSolution','penalty');
```

```
                                Max     Line search  Directional  First-order
 Iter F-count        f(x)   constraint   steplength   derivative   optimality Procedure
    0      25      1027.12       -0.065
    1      57       1014.8     -0.08785      0.00781         -507    1.11e+003
    2      86      1005.36     -0.08236       0.0625         -319    2.01e+003
    3     114      962.869     -0.07207        0.125         -282          785
    4     143      943.424     -0.06756       0.0625         -276    1.13e+003
    5     172      930.842     -0.06334       0.0625         -185          449
    6     201      906.952     -0.08349       0.0625         -204          869
    7     228      881.447     -0.06822         0.25         -140          700
    8     257      865.803     -0.06395       0.0625         -102    1.09e+003
    9     287      855.341     -0.07248       0.0313        -27.2          312
   10     315      829.506     -0.06431        0.125         -139          272
   11     345      820.504      -0.0623       0.0313        -61.6          173
   12     374      818.884     -0.07164       0.0625        -30.3          293
   13     404      817.852      -0.0694       0.0313        -53.5          197
   14     432      814.349     -0.06542        0.125        -70.8          234
   15     461      806.586     -0.06633       0.0625        -55.3          258
   16     490      801.971     -0.06219       0.0625        -35.8          578
   17     518      788.224     -0.06727        0.125        -28.6          139
   18     547      787.841      -0.0637       0.0625        -27.6          139
   19     575      785.354     -0.05834        0.125        -31.3          348
   20     607       784.56     -0.05864      0.00781          -14          329
   21     637      782.797     -0.06123       0.0313        -20.5          205
   22     666      776.538     -0.06852       0.0625        -43.2          129
   23     696       774.44     -0.06689       0.0313        -8.03         99.8
   24     726      773.729     -0.06601       0.0313        -10.1         48.5
   25     755      773.593     -0.06608       0.0625        -8.25         74.8
   26     785      773.477      -0.0676       0.0313        -16.8          145
   27     813       772.91     -0.06549        0.125        -13.3         83.8
   28     842       772.49     -0.06428       0.0625        -6.84         70.5
   29     875       772.46     -0.06441      0.00391        -3.68         64.1
   30     906      772.335     -0.06421       0.0156        -5.37         74.8
   31     935      771.619     -0.06562       0.0625        -13.7         32.8
   32     966      771.536     -0.06572       0.0156        -1.99         18.8
   33     995      771.412     -0.06585       0.0625        -2.67         18.2
   34    1023      771.283     -0.06621        0.125         -3.7         27.6
   35    1052      771.183     -0.06656       0.0625        -2.06         37.3
   36    1077      771.054     -0.06551            1        -4.33         13.8
   37    1102      770.949     -0.06566            1         -5.5         5.97
   38    1127      770.935     -0.06585            1       -0.967         3.12
   39    1152      770.934     -0.06578            1       -0.166        0.455
   40    1177      770.934     -0.06579            1      -0.0247        0.092
   41    1202      770.934     -0.06578            1      -0.0118        0.016  Hessian modified
   42    1227      770.934     -0.06579            1     -0.00481      0.00299  Hessian modified
Optimization terminated: directional derivative predicts change in
  objective value less than options.TolFun and maximum constraint violation
  is less than options.TolCon.
No active inequalities.
```

## Run Kalman smoother and projections.

A Kalman smoother yields the estimates of the unobservables based on the whole sample.

The output arguments of the `filter` function are

- `m1`, a model object with estimates of the unobservables
- `smooth`, a struct with `mean` and `std` databases containing the results of the Kalman smoother
- `se2`, the estimate of the overall variance scaling factor (when the `'relative'` option is set to true, which is by default)
- `pe`, a database with prediction errors for the measurement variables

The input arguments of the `filter` function are

- `mstar`, a model object to be filtered
- `d`, a database with measurement variables
- `starthist:endproj`, date range for filtration
- `'ahead'`,(=12), maximum number of steps ahead for which predictions of measurement variables are calculated

```
[m1,smooth,se2,delta,pe] = filter(mstar,d,starthist:endproj,'ahead',12);
f = smooth.mean;
f.std = smooth.std;
```

## Correct unit-root variables for an implicit constant.

```
%This is done to retrieve the correct level of unit root variables
%observed in growth rates

date = get(d.LGDP,'last');
if ~isnan(date)
    add = d.LGDP(date) - f.LGDP(date);
    f.LGDP = f.LGDP + add;
    f.LGDP_BAR = f.LGDP_BAR + add;
end
date = get(d.PIE4,'last');
if ~isnan(date)
    add = d.PIE4(date) - f.PIE4(date);
    f.PIE4 = f.PIE4 + add;
    f.PIELTE = f.PIELTE + add;
end
date = get(d.CAPU,'last');
if ~isnan(date)
    add = d.CAPU(date) - f.CAPU(date);
    f.CAPU = f.CAPU + add;
    f.CAPU_BAR = f.CAPU_BAR + add;
end
```

## Save results in a country- and version- specific .mat file.

```
dirs = {'Estimates'};
for ii = 1 : numel(dirs)
    if exist(sprintf('Results/Version%s/%s',version,dirs{ii}),'dir') ~= 7
        mkdir(sprintf('Results/Version%s/%s',version,dirs{ii}));
    end
end

filename = sprintf('Results/Version%s/Estimates/%s%s.mat',version,country,version);
fprintf('Saving %s.\n',filename);
save(filename,'f','d','mstar','Hess','Estar','E','starthist','p', 'pe');
```

```
Saving Results/VersionVER2/Estimates/USVER2.mat.
```

# printReport.m – creates a detailed report of the results

## Contents


- Clear workspace.
- Version to report. BE SURE TO MAKE CHANGES TO VERSION DESCRIPTION !!!
- Check for output directories.
- Create new empty report.
- Include model code listing.
- Add economic activity gaps graph.
- Add output gap graph.
- Add potential growth graph.
- Add YoY potential growth graph.
- Add unemployment rate graph.
- Add capacity utilization graph.
- Add relative variances graph.
- Create and include file with parameter estimate table.
- Create and include file with calibrated parameter table.
- Create and include file with RMSE table.
- Create and include file with revisions table.
- Add real-time graph.
- Add real-time HP graph.
- Compile PDF.
- Save output database as CSV.
- Housekeeping


## Clear workspace.

Read definitions of dates and the list of countries.

```
clear('all');
close('all');
home();

readDefinitions;

%add the path to functions used througout the file to do certain tasks
addpath Functions/
```

## Version to report. BE SURE TO MAKE CHANGES TO VERSION DESCRIPTION !!!

Each time you run `estimateParams`, the results are saved under a new version number. To access and report the correct results, you need to change the following line accordingly.

```
%version = 'VER0';
version = 'VER2';

fprintf('Version to report: %s.\n',version);
```

Version to report: VER2.

## Check for output directories.

The result files will be moved to `Reports`, `Graphs`, `OutputData` `TexFiles` directories. Make sure they exist.

```
dirs = {'Reports','Graphs','Reports','OutputData','TexFiles'};
for i = 1 : numel(dirs)
    if exist(sprintf('Results/Version%s/%s',version,dirs{i}),'dir') ~= 7
        mkdir(sprintf('Results/Version%s/%s',version,dirs{i}));
    end
end

fprintf('%s...\n',country);
load(sprintf('Results/Version%s/Estimates/%s%s.mat',version,country,version),...
    'f','d','mstar','Hess','Estar','E','starthist','p', 'pe');

gray = 0.3*[1,1,1];
generalstyle = struct();
generalstyle.axes.xgrid = 'on';
generalstyle.axes.ygrid = 'on';
generalstyle.line.lineWidth = 2.5;
generalstyle.line.color = {gray,'blue','red','green'};
generalstyle.line.lineStyle = {'-','-','-','-'};
generalstyle.patch.faceColor = [0.85,0.85,1];
generalstyle.patch.edgeColor = 'none';
generalstyle.bar.faceColor = 0.6*[1,1,1];
generalstyle.bar.edgeColor = 'none';
generalstyle.bar.barWidth = 1;
```

US...

## Create new empty report.

```
x = report.new();
```

## Include model code listing.

```
x.include('The Model','MODYUK.model','verbatim',true)

x.clearpage();
```

## Add economic activity gaps graph.

```
thisstyle = generalstyle;
thisstyle.line.color = {'blue',[0,0.5,0],'red'};
thisstyle.line.lineStyle = '-';

x.figure('Economic activity gaps',...
    'range',starthist:endproj,'saveas',sprintf('%sGraphEconomicActivityGaps%s',country,version));

x.graph('','legend',true,'zeroline',true,'style',thisstyle,'highlight',endhist+1:endproj);
x.series('Inflation',f.PIE4,'plotfunc',@bar);
x.series('Output gap',f.Y);
x.series('Unemployment gap',f.UNR_GAP);
x.series('Capacity utilisation gap',f.CAPU_GAP);

x.clearpage();
```

## Add output gap graph.

```
x.figure('Output Gap +/- 2 std devs (w/o model or parameter uncertainty)',...
    'range',starthist:endproj,'saveas',sprintf('%sGraphOutputGap%s',country,version));

x.graph('','legend',true,'highlight',endhist+1:endproj,'zeroline',true,'style',generalstyle);
x.series('Inflation',f.PIE4);
x.band('Output gap',f.Y,-2*f.std.Y,2*f.std.Y);

x.clearpage();
```

## Add potential growth graph.

```
x.figure('Output Growth (Q/Q Annualized) +/- 2 std devs (w/o model or parameter uncertainty)',...
    'range',starthist:endproj,'saveas',sprintf('%sGraphPotentialGrowth%s',country,version));

x.graph('','legend',true,'style',generalstyle,'highlight',endhist+1:endproj);
x.series('Actual growth',f.GROWTH);
x.band('Potential growth +/- 2 std devs',f.GROWTH_BAR,-2*f.std.GROWTH_BAR,2*f.std.GROWTH_BAR);
x.clearpage();
```

## Add YoY potential growth graph.

```
x.figure('Output Growth (Y/Y) +/- 2 std devs (w/o model or parameter uncertainty)',...
    'range',starthist:endproj,'saveas',sprintf('%sGraphPotentialGrowthYoY%s',country,version));

x.graph('','legend',true,'style',generalstyle,'highlight',endhist+1:endproj);
```

```
x.series('Actual',f.GROWTH4);
x.band('Potential +/- 2 std devs',f.GROWTH4_BAR,-2*f.std.GROWTH4_BAR,2*f.std.GROWTH4_BAR);
x.clearpage();
```

## Add unemployment rate graph.

```
x.figure('Unemployment Rate',...
    'range',starthist:endproj,'saveas',sprintf('%sGraphUnemploymentRate%s',country,version));

x.graph('','legend',true,'legendlocation','North','highlight',endhist+1:endproj,'style',generalstyle);
x.series('Actual ',f.UNR);
x.band('Equilibrium +/- 2 std devs',f.UNR_BAR,-2*f.std.UNR_BAR,2*f.std.UNR_BAR);
x.series('Steady-state unemployment rate',tseries(starthist:endproj,f.unr_ss));

x.clearpage();
```

## Add capacity utilization graph.

```
x.figure('Equilibrium Capacity Utilization +/- 2 std devs (w/o model or parameter uncertainty)',...
    'range',starthist:endproj,'saveas',sprintf('%sGraphCapacityUtilisation%s',country,version));

x.graph('','legend',true,'highlight',endhist+1:endproj,'style',generalstyle);
x.series('Actual',f.CAPU);
x.band('Equilibrium+/- 2 std devs',f.CAPU_BAR,-2*f.std.CAPU_BAR,2*f.std.CAPU_BAR);

x.clearpage();
```

## Add relative variances graph.

```
thisstyle = generalstyle;
thisstyle.line.color = {'blue','red'};

x.figure('Relative Variances of Potential Growth and Changes in the Output Gap at Different Horizons',...
    'range',1:20,'saveas',sprintf('%sGraphRelativeVariances%s',country,version));

[Rgap, Rpot] = computeRelativeVariances(f,starthist,endhist);
x.graph('','legend',true,'style',thisstyle);
x.series('Output gap',Rgap);
x.series('Potential output',Rpot);

x.clearpage();
```

### Create and include file with parameter estimate table.

```
createEstimTable(Hess,E,mstar,p);
x.include('Regularized Maximum Likelihood','estimTable');

x.clearpage();
```

### Create and include file with calibrated parameter table.

```
createCalibTable(mstar,Estar);
x.include('Calibrated Parameters','calibTable');

x.clearpage();
```

### Create and include file with RMSE table.

- `pe` is a structure containing prediction errors for each of the measurement variables and different step-aheads. It is created as output of the `filter` function in `estiamteParams.m`
- `vlist` is a cell of strings containing the variable names for which you want RMSEs computed and displayed. It must be a subset of the observables defined in the model (i.e. the fieldnames of the `pe` structure)

```
vlist = {'GROWTH_','PIE4_','UNR_'};
createRMSETable(pe,vlist);
x.include('RMSEs','RMSETable');

x.clearpage();
```

### Create and include file with revisions table.

```
[HPY0,Y0,targ_Y,targ_Y_hp] = createRevisionsTable(mstar,f,d,endhist);
x.include('Mean absolute revisions (according to most recent estimates)','RevisionsTable');
```

### Add real-time graph.

```
x.figure('Real-time output gap (MV)',...
    'range',starthist:endhist,'saveas',sprintf('%srealtimeY%s',country,version));

x.graph('','legend',true,'zeroline',true,'style',generalstyle);
```

```
x.series('Y (real-time)',Y0);
x.series('Y (final estimate)',targ_Y);

x.clearpage()
```

## Add real-time HP graph.

```
x.figure('Real-time output gap (HP)',...
    'range',starthist:endhist,'saveas',sprintf('%srealtimeYHP%s',country,version));

x.graph('','legend',true,'zeroline',true,'style',generalstyle);
x.series('Y (real-time)',HPY0);
x.series('Y (final estimate)',targ_Y_hp);

x.clearpage()
```

## Compile PDF.

```
x.publish(sprintf('%sReport%s.pdf',country,version),...
    'title',sprintf('ModYUK Report: %s%s',country,version),...
    'maketitle',true);
```

```
This is pdfTeX, Version 3.141592-1.40.7 (MiKTeX 2.7)
entering extended mode
(tp98d766b2_1242_4a91_8af2_89789d5d0abd.tex
LaTeX2e <2005/12/01>
Babel <v3.8j> and hyphenation patterns for english, dumylang, nohyphenation, ge
rman, ngerman, french, loaded.
(D:\Apps\Miktex2_7\tex\latex\base\article.cls
Document Class: article 2005/09/16 v1.4f Standard LaTeX document class
(D:\Apps\Miktex2_7\tex\latex\base\size10.clo))
(D:\Apps\Miktex2_7\tex\latex\psnfss\mathptmx.sty)
(D:\Apps\Miktex2_7\tex\latex\graphics\graphicx.sty
(D:\Apps\Miktex2_7\tex\latex\graphics\keyval.sty)
(D:\Apps\Miktex2_7\tex\latex\graphics\graphics.sty
(D:\Apps\Miktex2_7\tex\latex\graphics\trig.sty)
(D:\Apps\Miktex2_7\tex\latex\00miktex\graphics.cfg)
(D:\Apps\Miktex2_7\tex\latex\graphics\pdftex.def)))
(D:\Apps\Miktex2_7\tex\latex\sectsty\sectsty.sty)
(D:\Apps\Miktex2_7\tex\latex\graphics\color.sty
(D:\Apps\Miktex2_7\tex\latex\00miktex\color.cfg))
(D:\Apps\Miktex2_7\tex\latex\rotating\rotating.sty
(D:\Apps\Miktex2_7\tex\latex\base\ifthen.sty))
(D:\Apps\Miktex2_7\tex\latex\geometry\geometry.sty
```

```
(D:\Apps\Miktex2_7\tex\latex\geometry\geometry.cfg))
No file tp98d766b2_1242_4a91_8af2_89789d5d0abd.aux.
(D:\Apps\Miktex2_7\tex\latex\psnfss\ot1ptm.fd)
(D:\Apps\Miktex2_7\tex\context\base\supp-pdf.tex
[Loading MPS to PDF converter (version 2006.09.02).]
) (D:\Apps\Miktex2_7\tex\latex\psnfss\ot1ztmcm.fd)
(D:\Apps\Miktex2_7\tex\latex\psnfss\omlztmcm.fd)
(D:\Apps\Miktex2_7\tex\latex\psnfss\omsztmcm.fd)
(D:\Apps\Miktex2_7\tex\latex\psnfss\omxztmcm.fd) [1{C:/Documents and Settings/A
ll Users/Application Data/MiKTeX/2.7/pdftex/config/pdftex.map}] [2] [3]
<USGraphEconomicActivityGapsVER2.pdf, id=15, 516.93124pt x 638.385pt>
<use USGraphEconomicActivityGapsVER2.pdf> [4 <USGraphEconomicActivityGapsVER2.p
df>] <USGraphOutputGapVER2.pdf, id=24, 516.93124pt x 636.3775pt>
<use USGraphOutputGapVER2.pdf> [5 <USGraphOutputGapVER2.pdf>]
<USGraphPotentialGrowthVER2.pdf, id=33, 516.93124pt x 636.3775pt>
<use USGraphPotentialGrowthVER2.pdf> [6 <USGraphPotentialGrowthVER2.pdf>]
<USGraphPotentialGrowthYoYVER2.pdf, id=42, 516.93124pt x 636.3775pt>
<use USGraphPotentialGrowthYoYVER2.pdf> [7 <USGraphPotentialGrowthYoYVER2.pdf>]
 <USGraphUnemploymentRateVER2.pdf, id=52, 517.935pt x 636.3775pt>
<use USGraphUnemploymentRateVER2.pdf> [8 <USGraphUnemploymentRateVER2.pdf>]
<USGraphCapacityUtilisationVER2.pdf, id=61, 516.93124pt x 636.3775pt>
<use USGraphCapacityUtilisationVER2.pdf> [9 <USGraphCapacityUtilisationVER2.pdf
>] <USGraphRelativeVariancesVER2.pdf, id=70, 513.92pt x 640.3925pt>
<use USGraphRelativeVariancesVER2.pdf> [10 <USGraphRelativeVariancesVER2.pdf>]
(estimTable.tex) [11] (calibTable.tex) [12] (RMSETable.tex) [13]
(RevisionsTable.tex) <USrealtimeYVER2.pdf, id=89, 516.93124pt x 629.35126pt>
<use USrealtimeYVER2.pdf> [14] [15 <USrealtimeYVER2.pdf>]
<USrealtimeYHPVER2.pdf, id=103, 516.93124pt x 629.35126pt>
<use USrealtimeYHPVER2.pdf> [16 <USrealtimeYHPVER2.pdf>]
(tp98d766b2_1242_4a91_8af2_89789d5d0abd.aux) ){D:/Apps/Miktex2_7/fonts/enc/dvip
s/fontname/8r.enc}<D:/Apps/Miktex2_7/fonts/type1/bluesky/cm/cmmi10.pfb><D:/Apps
/Miktex2_7/fonts/type1/bluesky/cm/cmsy10.pfb><D:/Apps/Miktex2_7/fonts/type1/blu
esky/cm/cmtt10.pfb><D:/Apps/Miktex2_7/fonts/type1/urw/times/utmb8a.pfb><D:/Apps
/Miktex2_7/fonts/type1/urw/times/utmr8a.pfb><D:/Apps/Miktex2_7/fonts/type1/urw/
times/utmr8a.pfb><D:/Apps/Miktex2_7/fonts/type1/urw/times/utmri8a.pfb>
Output written on tp98d766b2_1242_4a91_8af2_89789d5d0abd.pdf (16 pages, 123981
bytes).
Transcript written on tp98d766b2_1242_4a91_8af2_89789d5d0abd.log.
```

## Save output database as CSV.

This database contains all variables including the estimates of the shocks and unobservables backed out by the Kalman filter.

```
f.std_UNR_BAR = f.std.UNR_BAR;
f.std_Y = f.std.Y;
f.std_G = f.std.G;
f.std_CAPU_BAR = f.std.CAPU_BAR;
```

```
f.std_UNR_BAR = f.std.UNR_BAR;
f.std_GROWTH_BAR = f.std.GROWTH_BAR;
f = rmfield(f,'std');
dbsave(f,sprintf('%sOutputData%s.csv',country,version),Inf,...
    'class',false,'comment',false,'format','%.16f');
```

## Housekeeping

```
movefile(sprintf('%sOutputData%s.csv',country,version),...
    sprintf('Results/Version%s/OutputData/',version));
movefile(sprintf('%sReport%s.pdf',country,version),...
    sprintf('Results/Version%s/Reports/',version));
movefile(sprintf('%sGraph*%s.*',country,version),...
    sprintf('Results/Version%s/Graphs/',version));
movefile(sprintf('%srealtime*%s.*',country,version),...
    sprintf('Results/Version%s/Graphs/',version));
movefile('estimTable.tex',...
    sprintf('Results/Version%s/TexFiles/%sestimTable%s.tex',version,country,version));
movefile('calibTable.tex',...
    sprintf('Results/Version%s/TexFiles/%scalibTable%s.tex',version,country,version));
movefile('RMSETable.tex',...
    sprintf('Results/Version%s/TexFiles/%sRMSETable%s.tex',version,country,version));
movefile('RevisionsTable.tex',...
    sprintf('Results/Version%s/TexFiles/%sRevisionsTable%s.tex',version,country,version));

fprintf('%s Done.\n',country);
```

```
US Done.
```