

Appendix for “Are variations in term premia related to the macroeconomy?”

*Gregory R. Duffee**

Haas School of Business
University of California – Berkeley

This Draft: June 22, 2007

1 Dear reader...

This appendix describes in detail the maximum likelihood estimation performed for the paper. It also discusses the Monte Carlo simulations summarized in the paper. My web site contains the data and Matlab code used for estimation and simulation. I am making the code available as an experiment. My goals are (1) to limit the number of people who pester me for the code and (2) to give others an opportunity to find bugs in my code and let me know about them. My fear is that I will get hordes of graduate students (not my own) asking me questions about how the code works rather than figuring it out for themselves.

This code may be freely used and distributed by anyone, to anyone, without my permission or knowledge. I do not support the use of the code. My subjective probability of a bug somewhere in this code is almost equal to one. If you find one (or if you think you’ve found one), please let me know, keeping in mind that what you view as a bug I might think of as a feature. My subjective probability of an important bug is much smaller, but is not quite zero. If you think you’ve found one of these, please let me know. If necessary, I will fix the code and you will earn a special place in my heart.

This code executes and produces the results in the paper (absent typos in the paper). If you cannot get the code to execute on a Wintel PC running Matlab with the optimization and statistics packages installed, do not tell me about your problem. I know the code executes, so either you are doing something wrong or Matlab performs differently on your machine than on mine. In either case, I am unwilling to solve the problem.

A quick glance at the code reveals that I am a lousy programmer. I have no doubt you could make my code run much faster and look prettier. But you do not need to tell me about it. I have learned to accept my failings as a programmer; you should as well.

And finally, none of these warnings apply to anyone who might be a potential referee of the paper!

2 What is in the zip file?

The zip file has three directories. The directory `matlabCode` contains `.m` files. These are either matlab functions or collections of code that can be executed at a Matlab prompt. The directory `matlabData` contains data used to estimate the models. The directory `matlabFiles` contains input and output files with (initial, final) parameters, standard errors, etc.

*Voice 510-642-1435, email duffee@haas.berkeley.edu. This appendix and associated files are maintained at <http://faculty.haas.berkeley.edu/duffee/>.

Note that this appendix uses a **typewriter-style** font to indicate a file name and an *italicized* font to indicate the name of a Matlab *.m* file, which contains code.

3 Data

There are raw data files and constructed data files. All are contained in `matlabData`.

3.1 Raw data

3.1.1 Inflation and output

I use the GDP price index constructed by the BEA. I do not recall when I downloaded the data from the NIPA web site. The file `gdp_price_index.dat` contains the values of the quarterly index from 1947Q1 through 2006Q1.

I use real GDP (chain-weighted 2000 dollars, seasonally adjusted, quarterly values at annual rates). Once again, I do not recall when I downloaded the data. The file `gdp_real.dat` contains the values from 1947Q1 through 2006Q1.

3.1.2 Quarterly riskfree rate

The three-month bond yield is taken from the CRSP riskfree rates file. It is the “average” yield, not the bid or the ask. The yield as of the last month of every quarter (from 1925Q4 through 2005Q4) is put in the file `crsp_riskfree_q.dat`.

3.1.3 ZCB bond yields with maturities of one year and greater

Fed economists construct a zero-coupon yield curve. See the working paper “The U.S. Treasury Yield Curve: 1961 to the Present,” by Refet Gurkaynak, Brian Sack, and Jonathan Wright, FEDS working paper series 2006-28. They have an Excel file with daily data. It is `feds200628.xls`.

3.2 Data transformations

The *.m* file `create_data` constructs inflation, the growth rate of real GDP, and pulls the zero-coupon bond yields I want from the Fed’s Excel file. Inflation, the growth rate of output, and the three-month bill yield are put into `macrodata.dat`. There are two different files with zero-coupon bond yields. One file contains yields on bonds with maturities of one, two, three, five, and seven years. The other has an additional column for the ten-year yield. The file names are `yield7data.dat` and `yield10data.dat`. Two related files contain maturities, in quarters. They are `mats_max7.dat` and `mats_max10.dat`.

4 ML estimation

4.1 Overview

An input file contains information about the location of the data, the manner in which the model is estimated, and the starting values for the optimization routine. The estimation results are placed in an output file. A control file, `matlabFiles/control.dat`, contains the names of the input and output files. Matlab code is used to read the control file, then read the input file, read the data, estimate the model, and place the results in the output file.

4.2 An input file example

Assume that `control.dat` is

```
Input ..\matlabFiles\late_onlyomega.in
Output ..\matlabFiles\late_onlyomega.out
```

This input file begins with

```
% Input file for estimation of dynamic term structure model
% April 2006
MACRO DATA FILE (must be 3 columns)
..\matlabData\macrodata.dat
INDEX OF SHORT RATE (which of the above columns contains the short rate?)
3
NUMBER OF MULTIPERIOD BOND YIELDS
6
BOND YIELD DATA FILE (above integer must equal number of columns)
..\matlabData\yield10data.dat
MATURITY DATA FILE (must be single column with maturities of above bonds)
..\matlabData\mats_max10.dat
LENGTH, IN YEARS, OF A PERIOD
0.25
INDEX OF FIRST OBS USED IN ESTIMATION
132
INDEX OF LAST OBS USED IN ESTIMATION (big num means to eof)
215
```

This information tells the Matlab code the name of the file with the macro data. The Matlab code used to estimate the model requires that there be three macro variables, including the short rate. The placement of the short rate among these variables is specified by “INDEX OF SHORT RATE.” There is no limit on the number of multiperiod bond yields used in estimation. The input file says there are six, contained in the indicated data file. The input file also tells the Matlab code where to find the maturities of these bonds. The code assumes yields are annualized, but does not make any assumption about the frequency

of the observed data. The input file contains the length of a period. With quarterly data, the length is 0.25.

The data files I constructed contain data spanning 1952Q2 through 2005Q4. But in the paper, the “full” sample begins with 1961Q2. When estimating the model over the full sample, I use observations 37 through 215. When estimating the model over the recent subsample 1985Q1 through 2005Q4, I use observations 132 through 215.

Skip ahead in the input file to the parameter description. This description tells the Matlab code which parameters are to be estimated and which are to be fixed to particular values. It also tells the Matlab code how to generate random starting values for the numerical optimization.

PARAM DESCRIPTION

```
% [--name--  --flag if free (1=yes)-- --initial value-- --std dev--]
```

```
%
```

```
% If a variable is free, it is estimated. The initial value is
% used as a starting value in estimation unless OLS starting values
% is set to one. In this case, the initial values below are overridden.
```

```
% The standard deviations are used to generate random initial values,
```

```
% used when the estimation routine begins at a randomly chosen
```

```
% initial value.
```

```
%
```

FUNDAMENTAL VAR K MATRIX (must be 3 x 3 x numlags)

```
Kf11  1          0.300000    0.000001
```

```
Kf21  1          0.000000    0.000001
```

```
Kf31  1          0.000000    0.000001
```

```
Kf12  1         -0.100000    0.000001
```

```
Kf22  1          0.500000    0.000001
```

```
Kf32  1          0.000000    0.000001
```

```
Kf13  1         -0.100000    0.000001
```

```
Kf23  1         -0.200000    0.000001
```

```
Kf33  1          0.600000    0.000001
```

FUNDAMENTAL VAR SIGMA MATRIX (must be 6)

```
Sf11  1          0.300000    0.000001
```

```
Sf21  1          0.300000    0.000001
```

```
Sf22  1          0.300000    0.000001
```

```
Sf31  1          0.300000    0.000001
```

```
Sf32  1          0.300000    0.000001
```

```
Sf33  1          0.300000    0.000001
```

RISK PREMIA VAR K MATRIX (must be 3 x 3)

```
Ku11  1          0.900000    0.400000
```

```
Ku21  1          0.000000    0.400000
```

```
Ku31  1          0.000000    0.400000
```

```
Ku12  1          0.000000    0.400000
```

```
Ku22  1          0.500000    0.400000
```

```
Ku32  1          0.000000    0.400000
```

Ku13	1	0.000000	0.400000
Ku23	1	0.000000	0.400000
Ku33	1	0.200000	0.400000
RISK PREMIA VAR SIGMA MATRIX (must be 6)			
Su11	1	0.010000	0.100000
Su21	1	0.000000	0.100000
Su22	1	0.010000	0.100000
Su31	1	0.000000	0.100000
Su32	1	0.000000	0.100000
Su33	1	0.010000	0.100000
RISK PREMIA CONSTANT TERMS (must be 3)			
L0u1	1	0.000000	0.100000
L0u2	1	0.000000	0.100000
L0u3	1	0.000000	0.100000
RISK PREMIA EFFECT OF FUNDAMENTAL VARIABLES (must be 3 x 3)			
L111	0	0.000000	0.200000
L121	0	0.000000	0.200000
L131	0	0.000000	0.200000
L112	0	0.000000	0.200000
L122	0	0.000000	0.200000
L132	0	0.000000	0.200000
L113	0	0.000000	0.200000
L123	0	0.000000	0.200000
L133	0	0.000000	0.200000
MEASUREMENT ERROR SIGMA (must be 4)			
ERRp	1	0.800000	0.005000
ERRg	1	1.800000	0.005000
ERRr	0	0.000000	0.005000
ERRb	1	0.070000	0.005000

First note what is not here. The vector μ_f is not estimated. As noted in the paper, it is fixed by the requirement that the model's unconditional mean of the macro factors equals the sample mean. The nine elements of K_f are K_{fij} , where i is the row and j is the column. The six elements of Σ_f are S_{fij} , where i is the row and j is the column. The elements of K_ω are K_{uij} and Σ_ω are S_{uij} . The three elements of λ_{0f} are $L0_{ui}$.

The nine elements of λ_{1f} are $L1_{ij}$. The second column is zero for these parameters, indicating that these parameters are not to be estimated. Instead, they are set equal to the values in the third column. These values are all zero. This means that the input file corresponds to the general null hypothesis that $\lambda_{1f} = 0$. The input files containing the phrase `onlyomega` set this matrix to zero and specify that the parameters are not free. The last set of parameters are the standard deviations of measurement errors. They are, in order, the SDs for the macro variables (in the order of the variables in the datafile of macro variables) and the SD for the multiperiod bond yields.

The fourth column contains standard deviations used with randomly generated starting values for the numerical optimization. This randomization is described in more detail in the

context of the remaining rows of the input file. These remaining rows are:

```
USE OLS STARTING VALUES? (1=yes, 0=no)
1
USE LATENT FACTORS? (1=yes, 0=no)
1
NUMBER OF LOOPS USED IN NUMERICAL ESTIMATION
100
NUMBER OF SIMPLEX CALLS PER LOOP
5
NUMBER OF DERIVATIVE-BASED CALLS PER LOOP
1
```

The OLS flag tells the Matlab code whether to override some of the initial parameters specified in the input file. If the OLS flag is one, the starting values for K_f and Σ_f are set to those from OLS estimation of the VAR(1).

The latent factor flag is used when estimating the standard macro-finance model. If it is zero, the Matlab code is told to ignore the presence of the term premia factors.

The number of loops used in estimation works as follows. If it is zero, the model is not estimated. Instead, the likelihood function is calculated using the initial parameters. If it is one, the model is estimated, beginning the numerical optimization at the parameters passed in the input file (unless the OLS flag is one).

If the number of loops n exceeds one, the model is estimated n times with n different starting values. The first estimation begins the numerical optimization at the parameters passed in the input file (again, unless the OLS flag is one). All other optimizations begin at random points. The distribution of a given parameter is normal with mean equal to the third column in the input file (again, unless the OLS flag is one) and standard deviation passed in the fourth column. Note that in this input file (and all of the others here), the standard deviations for K_f and Σ_f are tiny. Thus the randomization is really only over the parameters associated with risk premia and measurement error.

Finally, this file tells the Matlab code how to perform the numerical optimization. For this input file, estimation is done first with Simplex. After five rounds of Simplex (each round beginning at the point the previous round ended), estimation switches to a single round of a derivative-based method.

Once estimation is complete, the output file is written. It has a structure identical to the input file. The only differences are (a) the initial parameter values are replaced by the ML estimates; (b) the final two lines of the file report the ML value and a code indicating the status of the optimization routine at that ML value. Here they are for `late_onlyomega.out`.

```
LOG-LIKELIHOOD VALUE
4636.919080
OPTIMIZATION EXIT FLAG
-99
```

A flag of -2 indicates that the derivative-based search could not find a better direction. A flag of -99 means that the code was interrupted somewhere. In this case, no standard errors are produced.

Other differences are (c) the output file reports the derivative of the log-likelihood function with respect to all parameters (including free and fixed); (d) the output file reports standard errors for free parameters. These are calculated two ways. One way, which hardly ever produces an invertible matrix, is the robust version, which uses the outer product of first derivatives and the Hessian. The other way just uses the outer product of first derivatives, which is always invertible.

4.3 The code used to estimate the model

Given the control file, input file, and the name of an output file, the model is estimated by executing the following command at a Matlab prompt:

```
>> run_ml
```

The *.m* file *run_ml* is

```
clear
control_file = char('..\matlabFiles\control.dat');

printscreen = 1; % print some intermediate results to the screen
writetemp = 1; % save results in temp files in case of crash
mineigen_allowed = -1; % keep model away from region where mean
                    % term structure is explosively
                    % zig-zagging.
restrict_unconmean = 1; % sets likelihood to bad value if
                    % unconditional shape of the term structure
                    % is highly implausible -- either in level
                    % or slope.
kalman_tolerance = 1e-8; % used for fast Kalman recursion, when
                    % estimating with simplex.
display_simplex = 0; % display iterations in simplex? 1 = yes.
display_deriv = 0; %display iterations in deriv method? 1 = yes.

%-----
%
% Estimate model
%
%-----

[numbonds all_estimates loglike score exitflag ...
mean_hessian S_matrix T] = ...
    estimate_model(control_file, printscreen, ...
                    writetemp, mineigen_allowed, kalman_tolerance, ...
                    restrict_unconmean, display_simplex, display_deriv);
```

The run-time can be quite long when many starting values are used. Because it is hard to distinguish between a long run time and an infinite loop that is running amok, I allow intermediate results to be printed to the screen. This is done with the flag *printscreens*. Extremely detailed results from the optimization routines can be turned on with the flags *display_simplex* and *display_deriv*.

Another problem with long run times is accidental early termination. A flag that allows recovery from crashes is *writetemp*. I do not describe the recovery code here. Some information and the code is in the file *run_crashrecover*.

The variable *mineigen_allowed* sets the lower bound on the eigenvalues of the equivalent-martingale K_x . As described in the paper, I set this to minus one to keep the optimization routine out of a region that fits the particular bond yields used in estimation but implies wildly implausible behavior at intermediate, unobserved maturities. The flag *restrict_unconmean* imposes restrictions on the unconditional level and shape of the yield curve, as described in the paper. The flag *kalman_tolerance* is described in Section 5 below.

This code, when done, produces the output file. (The run-time output of the routine *estimate_model* is used for other purposes.)

4.4 Some detail about the numerical optimization

I use the Matlab optimization routines *fminsearch* (a Simplex routine) and *fminunc* (a derivative-based routine). The Simplex routine has upper limits on the number of function evaluations and the number of Simplex iterations. I set them equal to 5,000 and 4,000 respectively. The other tolerances for Simplex are set to 1×10^{-16} . The tolerances for *fminunc* are set to 1×10^{-10} . (There are also maximums for function evaluations and iterations with *fminunc*, but in practice they are never hit.)

The derivative-based optimization routine can use either numerical derivatives or analytic derivatives. In an earlier version of the paper I used numerical derivatives. Unfortunately, numerical precision interacts with the numerical optimization in a way that unavoidably creates errors in the calculation of the optimum. When using numerical derivatives, these errors are unacceptably large. Therefore the current code uses analytic first derivatives.

5 A detail about the Kalman filter algorithm

Here I use the notation and equations of Section 3 in Harvey's *Forecasting, structural time series models and the Kalman filter*. The Riccati equation for the error covariance matrix $P_{t+1|t}$ has a steady state solution if the transition equation implies stationary dynamics. (See Section 3.3 in Harvey.) The code *kalman_recursion* does the standard Kalman filter recursion (equation 3.2.4c in Harvey) for each observation t . The code *kalman_recursion_fast* does the recursion faster. Rather than using the standard recursion for every t , the "fast" code first checks if the $P_{t_0+1|t_0}$ is close to $P_{t_0|t_0-1}$. If they are sufficiently close, I assume that this value of $P_{t_0+1|t_0}$ is the steady state value for all future t , and the recursion step is skipped for $t_0 + 2, \dots$

I define “close” as

$$1' \left(\left| \text{vech}(P_{t_0+1|t_0}) - \text{vech}(P_{t_0|t_0-1}) \right| \right) < \textit{kalman_tolerance}$$

where $1'$ is a row of ones. The tolerance level is set in *run.ml* as described in a previous section. In practice, setting this to 10^{-8} works well, in the sense that convergence occurs fairly quickly and the approximate log-likelihood value is almost identical to the true log-likelihood value.

The “fast” version is used with Simplex optimization. I switch to the slower version when using the derivative-based optimization method.

6 Regression simulations

The *.m* file *display_montecarloRegs* produces simulation results for forecasting regressions under both the restrictive null hypothesis and the general null hypothesis. The comments in the file describe how to modify the file to generate specific simulations. The simulations are produced by executing this file at a Matlab prompt. The results are displayed to the screen.