MIDAS MatLab Toolbox*

Esben Hedegaard†

August 29, 2011

Contents

1 Introduction 2

1.1 The MIDAS model 2
1.2 Estimation 4
1.3 Components of the MIDAS model 4
1.4 A Note on Calendar Time 5

2 Data Classes and their Construction 5

2.1 MidasDataGenerator 5
2.2 MidasYData 6
2.3 MidasXData 6

3 Parameters 7

4 Weighting Schemes 7

5 Specifying the Conditional Moments 8

6 The Objective Function 10

7 The MidasModel class 10

8 Estimating the Model 11

8.1 Keeping Track of Parameters 12
8.2 How the MidasModel and MidasEstimator classes work 13

A Standard Errors 14

A.1 MLE standard errors (check assumptions!) 15

B Pseudo maximum likelihood using the normal likelihood function 15

B.1 The standard MIDAS model 15
B.2 The MIDAS model with skewness 17

*I would like to thank Eric Ghysels, Arthur Sinko, and Ross Valkanov who generously shared their code for several of their papers using MIDAS models, and some of the functionality in this toolbox is taken from their code. The main purpose of this toolbox is to make it easy to combine several weighting schemes to model both conditional variance and conditional skewness in a MIDAS framework.

†esben.hedegaard@stern.nyu.edu
1 Introduction

1.1 The MIDAS model

This toolbox handles the estimation of univariate MIDAS models. Some examples from this class of models are

**Example 1.** Ghysels et al. (2005) estimate a model where the conditional expected return in month $t$ is a linear function of the conditional variance of the return in month $t$, and where the conditional variance is given by a MIDAS estimate. The model is

$$ R_{t+1} = M_{t+1} (\mu + \gamma V^MIDAS_t + \varepsilon_{t+1}) $$

and the parameters $\mu, \gamma$ and $\theta$ are estimated jointly using MLE. $R_{t+1}$ denotes the monthly return in month $t+1$, $r_{t-i}$ denotes the daily return on date $t-i$, and $(w_1(\theta), \ldots, w_K(\theta))$ is a 'weighting function' where the weights sum to 1. Finally, $M_{t+1}$ is the number of trading days in month $t+1$, ensuring the return is in monthly values. Note that $V^MIDAS_t$ is an estimate of the conditional variance of daily returns over the next month, and the variance of $R_{t+1}$ is thus $M_{t+1}V^MIDAS_t$. $\gamma$ is interpreted as relative risk aversion, and $\mu$ is a *expected daily return* due to possible omitted risk factors: The expected monthly return is $M_{t+1}(\mu + \gamma V^MIDAS_t)$.

**Example 2.** In the same paper, Ghysels et al. (2005) also estimate a model where the MIDAS estimate of conditional variance is given by an 'asymmetric' model where past positive and negative returns can have different weights. The model is

$$ R_{t+1} = M_{t+1} (\mu + \gamma V^MIDAS_t + \varepsilon_{t+1}) $$

$$ \varepsilon_{t+1} \sim N(0, V^MIDAS_t) $$

$$ V^MIDAS_t = \phi \sum_{i=1}^{K} w_i(\theta)r^2_{t-i-1}(r_{t-i}>0) + (2 - \phi) \sum_{i=1}^{K} w_i(\theta^-)r^2_{t-i-1}(r_{t-i}<0) $$

and the parameters $\mu, \gamma$ and $\theta$ are estimated jointly using MLE. $R_{t+1}$ denotes the monthly return in month $t+1$, $r_{t-i}$ denotes the daily return on date $t-i$, and $(w_1(\theta), \ldots, w_K(\theta))$ is a 'weighting function' where the weights sum to 1. Finally, $M_{t+1}$ is the number of trading days in month $t+1$, ensuring the return is in monthly values. Note that $V^MIDAS_t$ is an estimate of the conditional variance of daily returns over the next month, and the variance of $R_{t+1}$ is thus $M_{t+1}V^MIDAS_t$. $\gamma$ is interpreted as relative risk aversion, and $\mu$ is a *expected daily return* due to possible omitted risk factors: The expected monthly return is $M_{t+1}(\mu + \gamma V^MIDAS_t)$. 

---

17

C MidasYData and MidasXData

C.1 MidasYData

C.1.1 Constructing a MidasYData

C.1.2 Additional examples

C.2 MidasXData

D Simulation Experiment

D.1 Simulating from the MIDAS Model

D.1.1 MIDAS Model with $\mu = 0$

D.1.2 MIDAS Model with $\mu > 0$

D.1.3 A Different Specification of Conditional Variance

D.2 The Simulation

D.3 Large Sample Properties

D.4 Small Sample Properties: 850 Months

D.5 Small Sample Properties: 450 Months

D.6 Large Sample Properties: Alternative Specification

D.7 Small Sample Properties, 850 Months: Alternative Specification

D.8 Small Sample Properties, 450 Months: Alternative Specification
Positive and negative daily returns now have different impacts on the estimate of conditional variance for future periods. If \( \phi = 1 \) and \( \theta^+ = \theta^- \), this gives the model in Example 1. As above, the variance of \( R_{t+1} \) is \( M_{t+1} V_{t+1}^{\text{MIDAS}} \), and the expected monthly return is \( M_{t+1} (\mu + \gamma V_{t+1}^{\text{MIDAS}}) \).

**Example 3.** Generalize the MIDAS model to the following form, where we write \( w x_t = \sum_{i=1}^K w_i x_{t-i} \):

\[
y_{t+1} = f \left( w_1(\theta_1)x_{t+1}^{11}, \ldots, w_1(\theta_1)x_{t+1}^{1l_1}, \ldots, w_k(\theta_k)x_{t+1}^{1k_1}, \ldots, w_k(\theta_k)x_{t+1}^{1k_l}, x_{t+1}^{\text{OLS}} \right) + \varepsilon_{t+1}
\]

For example, \( y_t \) will typically be monthly observations. \( k \) is the number of different weighting schemes used, and each weighting scheme \( i \) can load on \( l_i \) different regressors. Note that potentially the \( x^{ij} \)'s can be sampled at different frequencies (one can predict monthly observations using past, say, daily and monthly observations), and that each weighting scheme can have a different number of lags.

The models in Examples 1 and 2 are clearly special cases of this model. Additional examples of models of the form in Example 3 are

**Example 4.** Suppose the conditional expected return is a linear function of conditional variance and conditional skewness, that conditional variance is modeled using daily squared returns, and that conditional skewness is modeled using daily cubed returns. To use the same weighting scheme for conditional variance and conditional skewness, take \( k = 1 \) weighting scheme and \( l_1 = l_2 = 1 \) regressor for this one weighting scheme. The model is

\[
R_{t+1} = M_{t+1} \left( \mu + \gamma_1 V_{t+1}^{\text{MIDAS}} + \gamma_2 S_{t+1}^{\text{MIDAS}} + \varepsilon_{t+1} \right)
\]

\[
\varepsilon_{t+1} \sim SN(0, V_{t+1}^{\text{MIDAS}}, S_{t+1}^{\text{MIDAS}})
\]

\[
V_{t+1}^{\text{MIDAS}} = \sum_{i=1}^K w_i(\theta_1)r_{t+1}^2
\]

\[
S_{t+1}^{\text{MIDAS}} = \sum_{i=1}^K w_i(\theta_2)r_{t+1}^3
\]

where SN denotes the skew normal distribution.

**Example 5.** Suppose again the conditional expected return is a linear function of conditional variance and conditional skewness, that conditional variance is modeled using daily squared returns, and that conditional skewness is modeled using daily cubed returns. To use different weighting schemes for conditional variance and conditional skewness, take \( k = 2 \) weighting schemes and \( l_1 = l_2 = 1 \) regressor for each weighting scheme. The model is

\[
R_{t+1} = M_{t+1} \left( \mu + \gamma_1 V_{t+1}^{\text{MIDAS}} + \gamma_2 S_{t+1}^{\text{MIDAS}} + \varepsilon_{t+1} \right)
\]

\[
\varepsilon_{t+1} \sim SN(0, V_{t+1}^{\text{MIDAS}}, S_{t+1}^{\text{MIDAS}})
\]

\[
V_{t+1}^{\text{MIDAS}} = \sum_{i=1}^K w_{1,i}(\theta_1)r_{t+1}^2
\]

\[
S_{t+1}^{\text{MIDAS}} = \sum_{i=1}^K w_{2,i}(\theta_2)r_{t+1}^3
\]

The toolbox allows the user to easily construct the ‘pairs’ of weighting schemes and MIDAS regressors, and then specify the transformation \( f \).
1.2 Estimation

There are two main applications of the MIDAS model. First, it can be used to estimate the risk-return tradeoff as in the examples above. The model is then estimated using MLE.

Second, the MIDAS model can be used to forecast volatility. In this case $y_{t+1}$ in equation (7) will be a measure of realized volatility over the following month, $RV_{t,t+1}$, and $x_t$ will contain measures of past realized volatility. An example of such a model is

$$RV_{t,t+1} = \mu + \beta w(\theta)x_t + \varepsilon_{t+1},$$

(16)

where $x_t = (RV_{t-1}, \ldots, RV_{t-N})$ and $RV_{t-d}$ denotes the realized volatility on day $t - d$. The model may be estimated using nonlinear least squares (NLS), minimizing the sum of squared errors.

1.3 Components of the MIDAS model

To estimate a MIDAS model, the following parts must be specified:

1. The $y$ variable and one or more $x$ variables to use in the model.
2. One or more weighting schemes.
3. The combination of weighting schemes and regressors.
4. The transformation $f$, specifying the expected return
5. An objective function, such as the sum of squared errors, or a likelihood function.

Example 6. To give an idea of what the code to estimate a MIDAS model will look like, suppose daily returns and the corresponding dates are stored in the variables `returns` and `dates` (the dates in `dates` must be in MatLab datenum format). We wish to estimate the risk-return tradeoff with the MIDAS model in Example 7 using monthly returns on the LHS and 252 lags of daily squared returns with beta-polynomial weights on the RHS. The model is estimated using MLE, assuming the innovations are normally distributed.

```matlab
% Specify how data should be aggregated
lagLength = 252; % Use 252 days lags
periodType = 'monthly'; % Monthly model
freq = 1; % The y-data will be aggregated over 1 month
isSimpleReturns = true; % The returns are simple, not log
% This class makes sure the LHS and RHS of the MIDAS model are conformable
generator = MidasDataGenerator(dates, periodType, freq, lagLength, isSimpleReturns);

% Construct data classes for y- and x-variables
yData = MidasYData(generator);
xData = MidasXData(generator);

yData.setData(returns);
xData.setData(returns.^2);

% Construct weighting scheme
theta1 = MidasParameter('Theta1', -0.001, 0, -10, true);
theta2 = MidasParameter('Theta2', -0.0001, 0, -10, true);
weight = MidasExpWeights([theta1; theta2]);
% Specify the conditional variance as $V^\{\text{MIDAS}\} = w^*x_t = \sum_{i=1}^{N} w_i x_{t-i}$
V_R = weight * xData;
```
Let \( E = \mu + \gamma V \). Scale \( \mu \) and \( \gamma \) assuming 22 days per month.

\[
\mu = \text{MidasParameter('Mu', 0.5/22, 10, -10)};
\gamma = \text{MidasParameter('Gamma', 2/22, 10, -10)};
\]

\[
E_R = \mu + \gamma \times V_R;
\]

% Construct the objective function
llf = MidasNormalLLF;

% Put it all together into a model
model = MidasModel(yData, E_R, V_R, llf);

% Now estimate the model
% Sdm specifies the Spectral Density Matrix used to calculate robust standard errors
sdm = SdmWhite;
estimator = MidasEstimator(model, sdm);
results = estimator.estimate();

1.4 A Note on Calendar Time

The program allows for estimating the models in ‘calendar time,’ meaning that the conditional expected return and conditional variance are calculated for each month. Given that different months have a different number of trading days, one must be careful in scaling the estimates correctly. If returns are iid, both the mean and the variance of monthly returns will be proportional to the number of trading days in the month.

2 Data Classes and their Construction

One of the main purposes of MIDAS is to handle data recorded with different frequencies. I first describe how to construct and handle the data used for the MIDAS model. There are two data objects: MidasYData and MidasXData for the LHS and RHS variables, respectively. To ensure the data in MidasXData will match that in MidasYData, both classes use a MidasDataGenerator class.

2.1 MidasDataGenerator

The MidasDataGenerator determines how observations are aggregated on the LHS of the MIDAS equation, and how many lags are used on the RHS. It also ensures the LHS and RHS have the same number of observations. The class is constructed using the constructor

\[
\text{MidasDataGenerator(dates, periodType, freq, lagLength, isSimpleReturns)}
\]

where the input variables are:

- **dates**: \( N \times 1 \) vector of dates for which the observations are recorded. In MatLab datenum format.
- **periodType**: Can be ‘monthly,’ ‘weekly’ or ‘daily’. Specifies the type of period over which the aggregation for the LHS should be done.
- **freq**: Integer. Specifies the number of periods that should be aggregated, e.g. 1 month or 2 weeks.
- **lagLength**: Integer. The lag length used for constructing the \( x \) variable in the MIDAS regression.
• **isSimpleReturn**: True/False. Specifies whether the returns to be used will be simple returns or log returns (as this will influence how to aggregate them).

**Example**: To construct monthly returns, specify `periodType = 'monthly'` and `freq = 1`.
**Example**: To construct bi-weekly returns, let `periodType = 'weekly'` and `freq = 2`.
**Example**: To use a fixed aggregation interval of, say, 22 trading days, use `periodType = 'daily'` and `freq = 22`.

### 2.2 MidasYData

The **MidasYData** class stores the observations to be used on the LHS of the MIDAS regression. To construct an instance of the class and set the data use the two class methods

- **MidasYData(generator)** Constructor. Takes a `MidasDataGenerator` as input.
- **setData(data)** Sets the actual data. `data` is an \( N \times 1 \) vector, typically consisting of daily returns.

**Example**: To construct monthly aggregated returns to use on the LHS of the MIDAS equation, use the following code (it’s assumed you have already constructed a vector `dateNums` of dates in MatLab datenum format, and a vector `returns` of daily returns).

```matlab
lagLength = 252; periodType = 'monthly'; freq = 1; isSimpleReturns = true;
generator = MidasDataGenerator(dateNums, periodType, freq, lagLength, isSimpleReturns);
yData = MidasYData(generator);
yData.setData(returns);
```

The **MidasYData** class provides functionality to easily visualize the data, and verify that it has been constructed correctly. See Appendix C for details.

### 2.3 MidasXData

The **MidasXData** stores the variables to be used on the RHS of the MIDAS regression. To construct an instance of the class and set the data use the two class methods

- **MidasXData(generator)** Constructor. Takes a `MidasDataGenerator` as input.
- **setData(data)** Sets the actual data. `data` is an \( N \times 1 \) vector, typically consisting of daily returns.

**Example**: To use 252 lags of daily squared returns on the RHS of the MIDAS equation, use the following code (again it’s assumed you have already constructed a vector `dateNums` of dates in MatLab datenum format, and a vector `returns` of daily returns):

```matlab
lagLength = 252; periodType = 'monthly'; freq = 1; isSimpleReturns = true;
generator = MidasDataGenerator(dateNums, periodType, freq, lagLength, isSimpleReturns);
xData = MidasXData(generator);
xData.setData(returns.^2);
```
To use absolute returns instead, write

```java
xDataAbs = MidasXData(generator);
xDataAbs.setData(abs(returns));
```

The `MidasXData` class provides functionality to easily visualize the data, and verify that it has been constructed correctly. See Appendix [C] for details.

### 3 Parameters

The next step in constructing the MIDAS model is to specify the parameters of the model. The class `MidasParameter` helps with this. To construct a variable, use the constructor

```java
MidasParameter(name, theta, upperBound, lowerBound, calibrate)
```

where the input variables are:

- **name**: String specifying the name of the parameter, say ‘theta1.’
- **theta**: Double. Initial value of the parameter. Changed during calibration.
- **upperBound**: Double. The upper bound on the parameter.
- **lowerBound**: Double. The lower bound on the parameter.
- **calibrate**: True/False. Specifies whether to calibrate the parameter (true) or hold the parameter fixed (false).

You will use `MidasParameter` in the weight functions described next, as well as when specifying the expected return as a function of the variance.

### 4 Weighting Schemes

The following weighting schemes are available:

- **MidasExpWeights**: The exponential weights of order $Q$ and lag length $K$ are given by

  $$
  w_d(\theta) = \frac{\exp(\theta_1 d + \theta_2 d^2 + \cdots + \theta_Q d^Q)}{\sum_{i=1}^{K} \exp(\theta_1 i + \theta_2 i^2 + \cdots + \theta_Q i^Q)}, \quad d = 1, \ldots, K
  $$

  The constructor of `MidasExpWeights` takes as input a vector of `MidasParameters` which determines the order $Q$ (say 2) of the polynomial (see example below).

- **MidasBetaWeights**: For lag length $K$, the weights are given by

  $$
  w_d(\theta_1, \theta_2) = \frac{f(u_d, \theta_1, \theta_2)}{\sum_{i=1}^{K} f(u_i, \theta_1, \theta_2)}, \quad f(u_i, \theta_1, \theta_2) = u_i^{\theta_1 - 1}(1 - u_i)^{\theta_2 - 1}, \quad d = 1, \ldots, K
  $$

  where $u_1, \ldots, u_K \in (0, 1)$ are the points where $f$ is evaluated. Two options for $u_i$ are available: $u_i$ can be equally spaced over $(\varepsilon, 1 - \varepsilon)$ where $\varepsilon$ is the machine precision, or $u_i$ can be equally spaced over $(1/K, 1 - 1/K)$. In the latter case we say the lags are ‘offset.’ The latter specification is often more numerically stable.

**Example**: To use exponential polynomial weights of order 2 with 252 lags, write
theta1 = MidasParameter('Theta1', -0.01, -0.0001, -10, true);
theta2 = MidasParameter('Theta2', -0.01, -0.0001, -10, true);
expWeights = MidasExpWeights([theta1; theta2]);

Here, both parameters will be calibrated using an upper bound of $-0.0001$ and a lower bound of $-10$.

**Example:** To use beta polynomial weights with 252 lags, write

theta1 = MidasParameter('Theta1', 1, 20, 0.0001, true);
theta2 = MidasParameter('Theta2', 2, 20, 0.0001, true);
offsetLags = false;
betaWeights = MidasBetaWeights([theta1; theta2], offsetLags);

Both parameters will be calibrated using an upper bound of 20 and a lower bound of 0.0001.

**Example:** A common weight function is the beta polynomial with $\theta_1$ fixed at 1. This is accomplished by not calibrating $\theta_1$:

theta1 = MidasParameter('Theta1', 1, 0.0001, 20, false); % Do not calibrate theta1
theta2 = MidasParameter('Theta2', 2, 0.0001, 20, true);
offsetLags = false;
betaWeights = MidasBetaWeights([theta1; theta2], offsetLags);

## 5 Specifying the Conditional Moments

We are now ready to combine the weighting schemes and the data in the desired way. This is done by writing the code in a way similar to what you would do when writing the model on a piece of paper. The usage is most easily illustrated with some examples (in all examples it is assumed the user has already constructed the weight functions and data objects such as the $x$-data to use).

**Example 7.** Consider the standard MIDAS model from Example 1:

\[ V_t^M = \sum_{i=1}^{N} w_i r_{t-i}^2, \quad E(R_{t+1}) = M_{t+1} (\mu + \gamma V_t^M). \]

This can be set up the following way:

V_R = weights * xDataSquared;
mu = MidasParameter('Mu', 0.01/22, 10, -10, true); % Monthly value of 1%
gamma = MidasParameter('Gamma', 2, 10, -10, true); % Relative risk aversion of 2
E_R = mu + gamma * V_R;

First, the Midas estimate of conditional variance is defined by combining the weight function with past squared returns. Then the expected return is specified as a linear function of the conditional variance. Note that the expected return will be scaled to monthly values, but that you don’t explicitly specify this in the code. ○

**Example 8.** Extending the standard MIDAS model to include skewness, we have

\[ V_t^M = \sum_{i=1}^{N} w_i r_{t-i}^2, \quad S_t^M = \sum_{i=1}^{N} w_i r_{t-i}^3, \quad E(R_{t+1}) = M_{t+1} (\mu + \gamma_1 V_t^M + \gamma_2 S_t^M) \]

where we model both conditional variance and skewness, and let the conditional expected return be a linear function of both. This can be set up the following way:

V_R = varWeights * xDataSquared;
S_R = skewWeights * xDataCubed;
mu = MidasParameter('Mu', 0.01/22, 10, -10, true); % Monthly value of 1%
gamma1 = MidasParameter('Gamma1', 2, 10, -10, true);
gamma2 = MidasParameter('Gamma2', 2, 10, -10, true);
E_R = mu + gamma1 * V_R + gamma2 * S_R;
Now consider the asymmetric MIDAS model. There are three steps in constructing the model: The first step constructs two pairs of weights and data, the second step makes the variance a function of these two pairs, and the third step makes the expected return a function of the variance.

**Example 9.** The asymmetric Midas model is

\[ V^M_t = \phi \sum_{i=1}^{N} w_1, i r_{t-1,1}^2 (r_{t-1} > 0) + (2 - \phi) \sum_{i=1}^{N} w_2, i r_{t-1,1}^2 (r_{t-1} < 0), \]

\[ E(R_{t+1}) = M_{t+1} (\mu + \gamma V^M_t). \]

This can be set up the following way:

```plaintext
lagLength = 252;
periodType = 'monthly';
freq = 1;
isSimpleReturns = true;
generator = MidasDataGenerator(dateNums, periodType, freq, lagLength, isSimpleReturns);

% Construct squared returns times indicator for positive return
xDataSquaredPos = MidasXData(generator);
xDataSquaredPos.setData(returns.^2 .* (returns > 0));
% Construct squared returns times indicator for negative return
xDataSquaredNeg = MidasXData(generator);
xDataSquaredNeg.setData(returns.^2 * (returns < 0));

% Combine data with weighting schemes
wxPos = weightsPos * xDataSquaredPos;
xnPos = weightsNeg * xDataSquaredNeg;

% Specify conditional variance
phi = MidasParameter('Phi', 1, 10, -10, true);
V_R = phi * wxPos + (2-phi) * wxNeg;

% Specify conditional expected return
mu = MidasParameter('Mu', 0.01/22, 10, -10, true); % Monthly value of 1%
gamma = MidasParameter('Gamma', 2, 10, -10, true);
E_R = mu + gamma * V_R;
```

**Example 10.** Extending the asymmetric MIDAS model to include skewness, the model is

\[ V^M_t = \phi_1 \sum_{i=1}^{N} w_1, i r_{t-1,1}^2 (r_{t-1} > 0) + (2 - \phi_1) \sum_{i=1}^{N} w_2, i r_{t-1,1}^2 (r_{t-1} < 0) \]

\[ S^M_t = \phi_2 \sum_{i=1}^{N} w_3, i r_{t-1,1}^3 (r_{t-1} > 0) + (2 - \phi_2) \sum_{i=1}^{N} w_4, i r_{t-1,1}^3 (r_{t-1} < 0) \]

\[ E(R_{t+1}) = M_{t+1} (\mu + \gamma_1 V^M_t + \gamma_2 S^M_t) \]

This can be set up the following way:

```plaintext
wx1 = weightsSquaredPos * xDataSquaredPos;
xw2 = weightsSquaredNeg * xDataSquaredNeg;
xw3 = weightsCubedPos * xDataCubedPos;
xw4 = weightsCubedNeg * xDataCubedNeg;
```
phi1 = MidasParameter('Phi1', 1, 10, -10, true);
V_R = phi1 * wx1 + (2-phi1) * wx2;

phi2 = MidasParameter('Phi2', 1, 10, -10, true);
S_R = phi2 * wx3 + (2-phi2) * wx4;

mu = MidasParameter('Mu', 0.01/22, 10, -10, true); % Monthly value of 1%
gamma1 = MidasParameter('Gamma1', 2, 10, -10, true);
gamma2 = MidasParameter('Gamma2', 2, 10, -10, true);
E_R = mu + gamma1 * V_R + gamma2 * S_R;

Example 11. The above examples are likely to be over-parameterized. Suppose instead we use the same weights to model conditional variance and skewness:

\[ V_M = \sum_{i=1}^{N} w_i r_{t-i}^2, \quad S_M = \sum_{i=1}^{N} w_i r_{t-i}^3, \quad E(r_{t+1}) = M_{t+1} (\mu + \gamma_1 V_M + \gamma_2 S_M) \]

This is easily set up like this:

V_R = weights * xDataSquared;
S_R = weights * xDataCubed; % Use same weight function

6 The Objective Function

We can now specify the objective function. This can either be the sum of squared errors, or one of several likelihood functions (currently only the normal likelihood function is implemented). The following objective functions are implemented

- MidasNormalLLF: Likelihood function for the normal distribution.
- MidasSSEObjective: Objective function for minimizing the sum of squared errors.

To use the normal likelihood function, simply write

llf = MidasNormalLLF;

7 The MidasModel class

The MIDAS model in general consists of a LHS variable, an expected return, a conditional variance, and an objective function such as the normal likelihood function. The class MidasModel collects all of these things. It has the constructor

MidasModel(yData, E_R, V_R, obj)

where the input variables are

- yData: MidasYData. The data on the LHS of the MIDAS equation.
- E_R: The expected return.
- V_R: The conditional variance.
- obj: MidasObjective class, specifying the objective function.
8 Estimating the Model

The model is estimated using the class `MidasEstimator`. It has two methods:

- `MidasEstimator(midasModel, sdm)` Constructor. Takes an instance of `MidasModel` as input as well as a spectral density matrix (see Appendix A).
- `estimate(options, params)` Estimates the model. The `options` argument is passed on to the optimizer, allowing the user to easily change the desired tolerance, or to print out the iterations. The `params` is also optional and specifies the sequence the estimated parameters are reported in.

The following repeats Example 12, showing the code to construct a MIDAS model from scratch.

**Example 12.** Suppose daily returns and the corresponding dates (in MatLab datenum format) are stored in the variables `returns` and `dates`. We wish to estimate the risk-return tradeoff with the MIDAS model in Example 1, using monthly returns on the LHS and 252 lags of daily squared returns with exponential weights on the RHS. The model is estimated using MLE, assuming the innovations are normally distributed.

```matlab
% Specify how data should be aggregated
lagLength = 252;    % Use 252 days lags
periodType = 'monthly';  % Monthly model
freq = 1;           % The y-data will be aggregated over 1 month
isSimpleReturns = true;  % The returns are simple, not log
% This class makes sure the LHS and RHS of the MIDAS model are conformable
generator = MidasDataGenerator(dates, periodType, freq, lagLength, isSimpleReturns);

yData = MidasYData(generator);
xData = MidasXData(generator);

yData.setData(returns);
xData.setData(returns.^2);

% Construct weighting scheme
theta1 = MidasParameter('Theta1', -0.001, 0, -10, true);
theta2 = MidasParameter('Theta2', -0.0001, 0, -10, true);
weight = MidasExpWeights([theta1; theta2]);
% Specify the conditional variance as V^{MIDAS} = w*x_t = sum_{i=1}^N w_i x_{t-i}
V_R = weight * xData;

% Let E = mu + gamma V.
mu = MidasParameter('Mu', 0.01/22, 10, -10);  % Monthly value of 1%
gamma = MidasParameter('Gamma', 2, 10, -10);
E_R = mu + gamma * V_R;

% Construct the objective function
llf = MidasNormalLLF;

% Put it all together into a model
model = MidasModel(yData, E_R, V_R, llf);
```
Now estimate the model

sdm specifies the Spectral Density Matrix used to calculate robust standard errors

\[
sdm = \text{SdmWhite};
\]

\[
estimator = \text{MidasEstimator(model, sdm)};
\]

\[
results = estimator.estimate();
\]

Calling the method \texttt{estimate} returns an instance of the class \texttt{MidasModelEstimate} with the variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimates</td>
<td>Vector of estimates of the parameters.</td>
</tr>
<tr>
<td>ParameterNames</td>
<td>Vector of the parameter names.</td>
</tr>
<tr>
<td>Likelihood</td>
<td>Value of the objective function.</td>
</tr>
<tr>
<td>VRobust</td>
<td>Robust estimate of the variance matrix for the parameter estimates. (see Appendix A)</td>
</tr>
<tr>
<td>StdErrorRobust</td>
<td>Robust standard errors of the parameters.</td>
</tr>
<tr>
<td>TstatRobust</td>
<td>Robust t-statistics for the parameters.</td>
</tr>
<tr>
<td>WMleScore</td>
<td>MLE estimate of the variance matrix for the parameter estimates. (see Appendix A)</td>
</tr>
<tr>
<td>StdErrorMleScore</td>
<td>MLE standard errors of the parameters.</td>
</tr>
<tr>
<td>TstatMleScore</td>
<td>MLE t-statistics for the parameters.</td>
</tr>
<tr>
<td>VMleHessian</td>
<td>MLE estimate of the variance matrix for the parameter estimates. (see Appendix A)</td>
</tr>
<tr>
<td>StdErrorMleHessian</td>
<td>MLE standard errors of the parameters.</td>
</tr>
<tr>
<td>TstatMleHessian</td>
<td>MLE t-statistics for the parameters.</td>
</tr>
<tr>
<td>MidasModel</td>
<td>The \texttt{MidasModel} estimated.</td>
</tr>
</tbody>
</table>

### 8.1 Keeping Track of Parameters

As described above, the toolbox allows the user to combine multiple weight functions and transformations into one model. Potentially, the model can thus have a large number of parameters. We need to be able to interpret the output.

Calling the function \texttt{estimate} on the \texttt{MidasEstimator} returns an instance of the class \texttt{MidasModelEstimate}. This class has a variable \texttt{ParameterNames}, holding the names of all the parameters estimated. To estimate a model, write

\[
estimator = \text{MidasEstimator(model, sdm)};
\]

\[
results = estimator.estimate();
\]

Calling \texttt{results.ParameterNames} returns

\[
\text{ans} =
\]

\[
\text{Theta1}
\]

\[
\text{Theta2}
\]

\[
\text{Mu}
\]

\[
\text{Gamma}
\]

This way it’s easy to see which estimates correspond to which parameters.

If you are not happy with the sequence of the parameters, you can force the estimator to use a specific sequence by passing it to the \texttt{estimate} function:

\[
estimator = \text{MidasEstimator(model, sdm)};
\]

\[
results = estimator.estimate(options, \{\text{Mu Gamma Theta1 Theta2}\});
\]

Calling \texttt{results.ParameterNames} now returns
ans =
Mu
Gamma
Theta1
Theta2

8.2 How the MidasModel and MidasEstimator classes work

As described above, the user has great flexibility to construct weighting functions, as well as expressions for the conditional expected return and the conditional variance. All these functions involve parameters to be calibrated. This section briefly describes how the program keeps track of the parameters. The *MidasModel* class collects the conditional expected return and the variance, and in doing so figures out which parameters the model contains. Put simply, the *MidasModel* ‘asks’ the conditional expected return and the conditional variance for their parameters. In turn, the conditional variance ‘asks’ its members for their parameters, ending with the weight function. This way, all parameters are collected in the *MidasModel* class. Note that the conditional expected return and the conditional variance contain many of the same parameters: The conditional expected return is constructed from the conditional variance, so all parameters used in the conditional variance are included twice. Hence, the final step is to pick out the unique parameters. Once this is done, the model knows which parameters to calibrate.
A Standard Errors

Both the ML estimator and the NLS estimator are M-estimators. Let \( w_t = (y_t, x_t) \). An M-estimator is an extremum estimator where the objective function to be maximized has the following form

\[
Q_T(\theta) = -\frac{1}{T} \sum_{t=1}^{T} m(w_t; \theta).
\] (22)

For MLE, \( m(w_t; \theta) \) is the log-likelihood function for observation \( t \), and for NLS \( m(w_t; \theta) \) is the squared error for observation \( t \).

Let

\[
s(w_t; \theta) = \frac{\partial m(w_t; \theta)}{\partial \theta}
\] (23)

\[
H(w_t; \theta) = \frac{\partial^2 m(w_t; \theta)}{\partial \theta \partial \theta'}
\] (24)

We will call \( s(w_t; \theta) \) the score vector for observation \( t \) (not the same as the gradient of the objective function), and \( H(w_t; \theta) \) the Hessian for observation \( t \) (not the same as the Hessian of the objective function).

Under certain regularity conditions, the M-estimator \( \hat{\theta} \) is asymptotically normal:

\[
\sqrt{T}(\hat{\theta} - \theta_0) \xrightarrow{D} N(\theta_0, \text{Avar}(\hat{\theta})).
\] (25)

The covariance matrix returned in the output is \( \frac{1}{T} \text{Avar}(\hat{\theta}) \), as this is what will be used for calculating confidence regions and \( t \)-statistics. Indeed, for a univariate parameter, the \( t \)-statistic is

\[
\frac{\sqrt{T}(\hat{\theta} - \theta)}{\sqrt{\text{Avar}(\hat{\theta})}} = \frac{\hat{\theta} - \theta}{\sqrt{\frac{1}{T} \text{Avar}(\hat{\theta})}}.
\] (26)

The asymptotic variance is given by

\[
\text{Avar}(\hat{\theta}) = (E[H(w_t; \theta_0)])^{-1} \Sigma (E[H(w_t; \theta_0)])^{-1}
\] (27)

where \( \Sigma \) is the long-run variance matrix of the process \( s(w_t; \theta) \) in the sense that

\[
\frac{1}{\sqrt{T}} \sum_{t=1}^{T} s(w_t; \theta) \xrightarrow{D} N(0, \Sigma).
\] (28)

To estimate \( \Sigma \), we need the process \( s(w_t; \theta), t = 1, \ldots, T \) which is found by numerical differentiation. Once the process \( s(w_t; \theta_0) \) is obtained, the long-run variance matrix \( \Sigma \) (also known as the spectral density matrix evaluated at zero), can be calculated in a number of ways. The method used is determined by the input \( \text{sdm} \) which must be a child-class of \texttt{SpectralDensityMatrix}. The following child-classes are currently available:

- \texttt{SdmWhite} Calculates the spectral density matrix using White’s method (the outer product of the gradient).
- \texttt{SdmNeweyWest} Calculates the spectral density matrix using the Newey West weighting scheme.
- \texttt{SdmHansenHodrick} Calculates the spectral density matrix using the Hansen-Hodrick weighting scheme.

For the MLE we assume that the innovations are independent, but with different variance, so it is most natural to use White’s estimate.

The estimate of the expected Hessian for observation \( t, E[H(w_t; \theta_0)] \), is calculated as \( 1/T \) times the Hessian of the objective function. However, evaluating this is often difficult: The entries in the Hessian are often
huge, of the order $10^{10}$, and some simple estimates can be way off. This toolbox evaluates the Hessian using
the function \texttt{hessian} from the MatLab Toolbox DERIVESTsuite. This seems very reliable, and also returns an estimated upper bound on the error of each of the second partial derivatives.

The code for calculating the robust variance estimate is this:

```matlab
H = hessian(@(a) estimator.MidasObj.val(a), results.aopt); % Calculate the Hessian
g = estimator.MidasObj.gradientProcessNumerical(results.aopt); % Calculate the gradient process
S = estimator.Sdm.calc(g); % Find the spectral density matrix for the gradient process
T = size(g,1);
% H above is the Hessian for the obj function. H/T is the Hessian for one obs.
Hinv = inv(H / T);
VRobust = 1 / T * Hinv * S * Hinv; % Estimate of asymptotic variance, scaled by 1/T
```

The robust variance estimate is returned in the output structure as \texttt{VRobust}, and the corresponding standard errors and $t$-statistics are returned in \texttt{stdErrorRobust} and \texttt{tstatRobust}.

### A.1 MLE standard errors (check assumptions!)

Importantly, when estimating the model using MLE, the usual MLE standard errors are not correct. The reason is that the observations in the MIDAS model are generally not iid: For instance, when using monthly returns on the LHS and one year of lagged daily returns on the RHS, the RHS observations are clearly not independent. As a consequence, the ‘information matrix equality’

$$E(H(w_t; \theta)) = -E(s(w_t; \theta)s(w_t; \theta)')$$

(29)

does not hold. Note that if the information matrix equality did hold, the asymptotic variance simplifies to

$$Avar(\hat{\theta}) = -(E[H(w_t; \theta_0)])^{-1} = (E[s(w_t; \theta)s(w_t; \theta)'])^{-1}.$$ (30)

These two estimates of the asymptotic variance are reported in \texttt{results.VmleHessian} and \texttt{results.VmleScore}. The corresponding standard errors are reported in \texttt{stdErrorMleHessian} and \texttt{stdErrorMleScore}, and the corresponding $t$-statistics are reported in \texttt{tstatMleHessian} and \texttt{tstatMleScore}.

### B Pseudo maximum likelihood using the normal likelihood function

#### B.1 The standard MIDAS model

This section considers a special application of the normal likelihood function. Suppose we do not know the true distribution of the error term $\varepsilon_{t+1}$ in the MIDAS model in Example 1:

$$R_{t+1} = \mu + \gamma V^M_t + \varepsilon_{t+1}. \tag{31}$$

However, assume we impose the restrictions $E_t(\varepsilon_{t+1}) = 0$ and $V_t(\varepsilon_{t+1}) = V^M_t$. As shown in the following this model can be estimated using GMM, where the moment conditions are given by the score function from the normal likelihood function. In other words, the model can be estimated consistently using the normal likelihood function, even though the error term is not assumed normal.

Let $\phi$ denote the density for the standard normal distribution. The score function for a single observation based on the normal distribution is

$$s(w_t; \theta) = \nabla_\theta \log(\phi(w_t; \theta)) = \frac{1}{\phi(w_t; \theta)} \nabla_\theta \phi(w_t; \theta). \tag{32}$$
Now, the density will be a function of the MIDAS estimates of the mean and variance:

\[
\phi(w_t; \theta) = \frac{1}{\sqrt{2\pi V_t^M(\kappa)}} \exp \left( \frac{-(R_{t+1} - \mu - \gamma V_t^M(\kappa))^2}{2V_t^M(\kappa)} \right) \tag{33}
\]

Differentiating wrt. \( \mu \) and \( \gamma \) we get

\[
\frac{\partial \phi(w_t; \theta)}{\partial \mu} = \frac{1}{\sqrt{2\pi V_t^M(\kappa)}} \exp \left( \frac{-(R_{t+1} - \mu - \gamma V_t^M(\kappa))^2}{2V_t^M(\kappa)} \right) \frac{R_{t+1} - \mu - \gamma V_t^M(\kappa)}{V_t^M(\kappa)} \tag{34}
\]

\[
= \phi(w_t; \theta) \frac{R_{t+1} - \mu - \gamma V_t^M(\kappa)}{V_t^M(\kappa)} - \phi(w_t; \theta) \varepsilon_{t+1} \tag{35}
\]

\[
\frac{\partial \phi(w_t; \theta)}{\partial \gamma} = \frac{1}{\sqrt{2\pi V_t^M(\kappa)}} \exp \left( \frac{-(R_{t+1} - \mu - \gamma V_t^M(\kappa))^2}{2V_t^M(\kappa)} \right) \left( R_{t+1} - \mu - \gamma V_t^M(\kappa) \right) \tag{36}
\]

\[
= \phi(w_t; \theta) \left( R_{t+1} - \mu - \gamma V_t^M(\kappa) \right) = \phi(w_t; \theta) \varepsilon_{t+1} \tag{37}
\]

Assuming only \( E_t(\varepsilon_{t+1}) = 0 \) without specifying the entire distribution of \( \varepsilon_{t+1} \), the expectation of the score is

\[
E \left( \frac{\partial}{\partial \mu} \log(\phi(w_t; \theta)) \right) = E \left( \frac{1}{\phi(w_t; \theta)} \phi(w_t; \theta) \varepsilon_{t+1} \right) = E \left( \varepsilon_{t+1} \right) = 0 \tag{38}
\]

\[
E \left( \frac{\partial}{\partial \gamma} \log(\phi(w_t; \theta)) \right) = E \left( \frac{1}{\phi(w_t; \theta)} \phi(w_t; \theta) \varepsilon_{t+1} \right) = E(\varepsilon_{t+1}) = E(E(\varepsilon_{t+1})) = 0 \tag{39}
\]

Similarly, differentiating wrt. \( \kappa \) we get

\[
\frac{\partial \phi(w_t; \theta)}{\partial \kappa} = \frac{1}{\sqrt{2\pi V_t^M(\kappa)}} \exp \left( \frac{-(R_{t+1} - \mu - \gamma V_t^M(\kappa))^2}{2V_t^M(\kappa)} \right) \tag{40}
\]

\[
+ \frac{1}{\sqrt{2\pi V_t^M(\kappa)}} \exp \left( \frac{-(R_{t+1} - \mu - \gamma V_t^M(\kappa))^2}{2V_t^M(\kappa)} \right) \left( 2(R_{t+1} - \mu - \gamma V_t^M(\kappa)) \gamma \frac{\partial}{\partial \kappa} V_t^M(\kappa) + (R_{t+1} - \mu - \gamma V_t^M(\kappa))^2 / 2 \frac{\partial}{\partial \kappa} V_t^M(\kappa) \right) \tag{41}
\]

\[
= \phi(w_t; \theta) \left( -\frac{\partial}{\partial \kappa} V_t^M(\kappa) \frac{1}{2V_t^M(\kappa)} \right) + \phi(w_t; \theta) \left( \frac{\varepsilon_{t+1} \gamma \frac{\partial}{\partial \kappa} V_t^M(\kappa)}{V_t^M(\kappa)} + \frac{\varepsilon_{t+1}^2 \frac{\partial}{\partial \kappa} V_t^M(\kappa)}{2(V_t^M)^2} \right) \tag{42}
\]

Assuming only \( E_t(\varepsilon_{t+1}) = 0 \) and \( V_t(\varepsilon_{t+1}) = V_t^M \) without specifying the entire distribution of \( \varepsilon_{t+1} \), the expectation of the score is

\[
E \left( \frac{\partial}{\partial \kappa} \log(\phi(w_t; \theta)) \right) = -\frac{\partial}{\partial \kappa} V_t^M(\kappa) \frac{1}{2V_t^M(\kappa)} + E \left( \frac{\varepsilon_{t+1} \gamma \frac{\partial}{\partial \kappa} V_t^M(\kappa)}{V_t^M(\kappa)} + E \left( \frac{\varepsilon_{t+1}^2 \frac{\partial}{\partial \kappa} V_t^M(\kappa)}{2(V_t^M)^2} \right) \right) \tag{43}
\]

\[
= -\frac{\partial}{\partial \kappa} V_t^M(\kappa) \frac{1}{2V_t^M(\kappa)} + \frac{V_t^M(\kappa) \frac{\partial}{\partial \kappa} V_t^M(\kappa)}{2(V_t^M)^2} = 0. \tag{44}
\]

This shows that it is possible to consistently estimate the parameters of the standard MIDAS model in Example 1 without making assumptions about the distribution of \( \varepsilon_{t+1} \).
B.2 The MIDAS model with skewness

Perhaps more interestingly, we can use the same approach to estimate the parameters in a MIDAS model where the expected return also depends on conditional skewness. Consider the model

\[ R_{t+1} = \mu + \gamma_1 V_t^M + \gamma_2 S_t^M + \varepsilon_{t+1} \]  

(47)

and assume only that \( E_t(\varepsilon_{t+1}) = 0 \) and \( V_t(\varepsilon_{t+1}) = V_t^M \). Let the MIDAS estimate of skewness be a function of the parameter \( \psi \). Using the likelihood function based on a normal distribution for \( \varepsilon_{t+1} \), we have

\[
\frac{\partial \phi(w_t; \theta)}{\partial \psi} = \frac{1}{\sqrt{2\pi V_t^M(\kappa)}} \exp \left( -\frac{(R_{t+1} - \mu - \gamma_1 V_t^M(\kappa) - \gamma_2 S_t^M(\psi))^2}{2V_t^M(\kappa)} \right) \times \frac{R_{t+1} - \mu - \gamma_1 V_t^M(\kappa) - \gamma_2 S_t^M(\psi)}{V_t^M(\kappa)} \left( -\gamma_2 \frac{\partial}{\partial \psi} S_t^M(\psi) \right) \\
= \phi(w_t; \theta) \frac{R_{t+1} - \mu - \gamma_1 V_t^M(\kappa) - \gamma_2 S_t^M(\psi)}{V_t^M(\kappa)} \left( -\gamma_2 \frac{\partial}{\partial \psi} S_t^M(\psi) \right) \\
= \phi(w_t; \theta) \frac{\varepsilon_{t+1}}{V_t^M(\kappa)} \left( -\gamma_2 \frac{\partial}{\partial \psi} S_t^M(\psi) \right)
\]

(48)

(49)

(50)

and taking the expectation of the score gives

\[
E \left( \frac{\partial}{\partial \psi} \log(\phi(w; \theta)) \right) = E \left( \frac{1}{\phi(w; \theta)} \frac{\partial}{\partial \psi} \phi(w_t; \theta) \right) = 0.
\]

(52)

Again, we can use the normal likelihood function to get consistent estimates of the parameters, even though in this case the error term \( \varepsilon_{t+1} \) is clearly not normally distributed—the model assumes \( \text{Skew}(\varepsilon_{t+1}) = S_t^M \). This is sometimes called quasi maximum likelihood estimation (QMLE). As the likelihood function is mis-specified, only the robust variance estimate should be used.

C MidasYData and MidasXData

C.1 MidasYData

The MidasYData class has the following properties:

- **Generator** The MidasDataGenerator used.
- **y** \( N \times 1 \) vector of observations.
- **Dates** \( N \times 1 \) vector of dates for which the observations are recorded. In MatLab datenum format.
- **EndOfLastPeriod** \( N \times 1 \) vector of dates denoting the end of the period previous to the recording of \( y \). In MatLab datenum format.
- **DateRange** \( N \times 1 \) vector where each entry contains the field range, e.g. DateRange(1).range. This range field is a vector of dates covered by the observations in \( y \).
- **PeriodLength** \( N \times 1 \) vector with the number of daily observations used to calculate the aggregated observations in \( y \).
- **NoObs** Number of observations in \( y \).

For example, \( y \) could contain monthly returns starting January 2000. **Dates** would contain the dates for Jan 31, 2000, Feb 28, 2000, and so on (more precisely, the last trading date of each month in datenum format). **EndOfLastPeriod** would be Dec 31, 1999; Jan 31, 2000 and so on, because Dec 31, 1999, is the end of the period immediately before the period for the first observation in \( y \) (it would really be the last trading
day in December). \texttt{DateRange(1).range} would contain the trading days in Jan 2000, \texttt{DateRange(2).range} would contain the trading days in Feb 2000 and so on (so the entries in \texttt{DateRange} do not have the same length). \texttt{RangeLength} contains the number of trading days in each month, and finally \texttt{NoObs} is the number of monthly returns in \texttt{y}.

The variable \texttt{EndOfLastPeriod} is used later to construct the RHS variable \texttt{x} of the MIDAS regression: to do this we must pick the \texttt{K} previous trading days for each return in \texttt{y}, where \texttt{K} is the lag length. \texttt{DateRange} is included so the user can verify how the data were constructed. \texttt{RangeLength} can be used to scale the MIDAS forecast of conditional variance: If one month has 25 trading days and another has 22, we might want to scale the forecast by 25 and 22, respectively.

The \texttt{MidasYData} class has the following methods, in addition to methods for accessing the properties (which are accessed by writing e.g. \texttt{yData.y()} where \texttt{yData} is an instance of \texttt{MidasYData}).

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{viewSummaryY()}</td>
<td>Prints the first and last 5 returns in \texttt{y}</td>
</tr>
<tr>
<td>\texttt{viewSummaryDate()}</td>
<td>Prints the first and last 5 dates in \texttt{Dates}</td>
</tr>
<tr>
<td>\texttt{viewSummaryDateRange()}</td>
<td>Prints the first and last 5 rows, and the first and last 3 columns of the dates in \texttt{DateRange}, along with the number of returns in each row (that is, the values in \texttt{PeriodLength}).</td>
</tr>
</tbody>
</table>

The purpose of the methods \texttt{viewSummaryY}, \texttt{viewSummaryDate} and \texttt{viewSummaryDateRange} is to make it easy to get a visual summary of the data to make sure it contains what it’s supposed to contain.

### C.1.1 Constructing a \texttt{MidasYData}

In this example we download daily returns from CRSP. We want to construct model using squared daily returns to forecast the variance of monthly returns, using a lag-length of 252 days. The data from CRSP look like this:

<table>
<thead>
<tr>
<th>Date</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>19280103</td>
<td>0.004444</td>
</tr>
<tr>
<td>19280104</td>
<td>-0.001197</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20001228</td>
<td>0.009421</td>
</tr>
<tr>
<td>20001229</td>
<td>-0.011276</td>
</tr>
</tbody>
</table>

The dates must first be converted to the MatLab datenum-format. We then construct a \texttt{MidasDataGenerator}:

```matlab
periodType = 'monthly';
freq = 1;
lagLength = 252;
isSimpleReturns = true;
generator = MidasDataGenerator(dnum, periodType, freq, lagLength, isSimpleReturns);
```

Next, we can use the \texttt{generator} to construct a \texttt{MidasYData}:

```matlab
yData = MidasYData(generator);
yData.setData(returns);
```

Using the methods \texttt{NoObs()}, \texttt{viewSummaryY()}, \texttt{viewSummaryDate()} and \texttt{viewSummaryDateRange()} the output is

- \texttt{NoObs()} = 864
- \texttt{ViewSummary()}
Note the following:

1. Because we need 252 daily returns to forecast the next month’s return, the `MidasDataGenerator` starts the monthly return in December 1928, although the data set starts in January 1928. As we’ll see below when looking at `MidasXData`, this is the first months for which we have 252 prior daily returns.

2. Even though the daily data continue to the end of December 2000, the last monthly return is for November. The reason is that the code looks for changes in the month (or week, or day) and does not see the change to January 2001.

3. As shown in the last section of the output, the first return (for 19281231) is calculated by aggregating the returns on 19281201, 19281203, ... , 19281231, a total of 25 daily returns.
C.1.2 Additional examples

The file `MidasYDataDemo` also contains examples of constructing data sets for forecasting weekly, bi-weekly, and 10-day observations.

C.2 MidasXData

Continuing the above example, I now demonstrate how to use the `MidasDataGenerator` to construct a `MidasXData`:

```matlab
xData = MidasXData(generator);
xData.setData(returns.^2);
```

Using the functions `NoObs()`, `viewSummaryX()` and `viewSummaryDateMatrix()` the output is

```
NoObs()
864
viewSummaryX()
Obs 1  2   3   ...   250  251  252
1  1.2574e-005  1.1451e-005  1.2299e-005  2.8016e-005  1.1415e-004  1.1903e-005
2  1.8295e-004  4.2107e-006  1.7873e-004  2.1800e-005  3.3856e-006  1.1903e-005
4  9.9222e-005  1.0609e-004  1.4493e-005  1.1594e-005  5.2563e-005  1.7245e-004
5  2.9046e-004  1.0695e-003  1.5969e-004  4.2477e-006  2.9604e-005  3.1784e-004
... 860  5.1999e-005  4.8651e-005  5.8003e-005  1.0201e-008  5.1348e-006  3.1784e-004
861  1.0897e-004  5.4995e-004  1.0435e-004  3.2365e-005  1.9980e-004  6.1009e-008
862  1.4378e-004  7.0896e-007  2.8516e-007  2.3581e-005  7.1129e-006  7.6292e-004
863  1.5361e-004  5.1784e-004  6.0812e-006  3.1343e-004  7.9032e-007  2.7852e-004
864  8.0514e-004  4.5064e-005  7.4391e-005  5.4676e-005  4.2903e-005  5.1912e-005
viewSummaryDateMatrix()
Obs 1  2   3   ...   250  251  252
1  19281130  19281128  19281127  19280126  19280125  19280124
2  19281231  19281229  19281228  19280227  19280225  19280224
3  19290131  19290130  19290129  19280328  19280327  19280326
4  19290228  19290227  19290226  19280424  19280423  19280420
5  19290328  19290327  19290326  19280525  19280524  19280523
... 860  20000630  20000629  20000628  19990708  19990707  19990706
861  20000731  20000728  20000727  19990805  19990804  19990803
862  20000831  20000830  20000829  19990908  19990907  19990906
863  20000929  20000928  20000927  19991006  19991005  19991004
864  20001031  20001030  20001027  19991105  19991104  19991103
```

The first return in the y-data is for 19281231 (see the previous section). The output shows that the 252 daily squared returns used to forecast that observation cover 19281130, . . . , 19280124. That is, we use returns ending in November to forecast the observation for December.
D Simulation Experiment

To verify the calculation of both the point estimates and standard errors, I perform a simulation experiment for the standard MIDAS model in Example 1. The model is

\[ R_{t+1} = \mu + \gamma V^{\text{MIDAS}}_{t+1} + \varepsilon_{t+1} \]  
\[ \varepsilon_{t+1} \sim N(0, V^{\text{MIDAS}}_t) \]  
\[ V^{\text{MIDAS}}_t = M_t \sum_{i=1}^{K} w_i(\theta) r^2_{t-i} \]  

D.1 Simulating from the MIDAS Model

D.1.1 MIDAS Model with \( \mu = 0 \)

When simulating from the MIDAS model with \( \mu = 0 \), the resulting returns have the same unconditional mean and variance, but their distribution changes over time. In fact, although the unconditional variance of \( r_t \) is the same for all \( t \), the distribution of \( r_t \) becomes increasingly peaked as \( t \) grows large. This makes simulating from the MIDAS model very complicated: Most simulated paths will ‘die out’ as the distribution of returns becomes more peaked around zero, and once in a while a tail-event occurs and the variance blows up.

To see this, consider the simplest example of a MIDAS model: The variance is estimated based on 1 past return, and is used to describe only the next day. For simplicity, consider the case where \( \mu = \gamma = 0 \). This model is thus, with \( r_t \) denoting daily returns,

\[ r_{t+1} = \varepsilon_{t+1} \sqrt{V^{\text{MIDAS}}_t} \]  
\[ \varepsilon_{t+1} \sim N(0, 1) \]  
\[ V^{\text{MIDAS}}_t = r^2_t \]  

Let \( \sigma^2_0 = V^{\text{MIDAS}}_0 \) be given. We can the explicitly write out the sequence of returns as

\[ r_1 = \varepsilon_1 \sigma_0 \]  
\[ \sigma^2_1 = r^2_1 = \varepsilon^2_1 \sigma^2_0 \]  
\[ r_2 = \varepsilon_2 \sigma_1 = \varepsilon_2 |\varepsilon_1 \sigma_0| \]  
\[ \sigma^2_2 = r^2_2 = \varepsilon^2_2 \sigma^2_0 \]  
\[ r_3 = \varepsilon_3 \sigma_2 = \varepsilon_3 |\varepsilon_2 \varepsilon_1 \sigma_0| \]  

Clearly, we will have \( r_t = \varepsilon_t \prod_{s=1}^{t-1} |\varepsilon_s| \sigma_0 \). As the \( \varepsilon_t \)'s are independent, \( E(r_t) = 0 \) for all \( t \), and \( V(r_t) = E(r^2_t) = \sigma^2_0 \). This shows that all returns have the same unconditional mean and variance. However, the distribution of \( r_t \) clearly changes with \( t \). Figure 1 shows the density of \( r_2, r_3, r_4, \) and \( r_5 \) when \( \sigma_0 = 1 \), as well as a typical simulated path for \( r_t \). Clearly the distribution quickly becomes very peaked in the top plot, and the simulated path for \( r_t \) dies out after just a few time-periods in the bottom plot. This effect also occurs when the variance estimate is based on a long history of returns, and when the variance is used to model a period, such as a month, of future returns.

D.1.2 MIDAS Model with \( \mu > 0 \)

With \( \mu > 0 \), the variance of the returns will no longer be constant over time—instead, it will grow as a function of \( t \). Despite this, it’s actually easier to simulate from this specification: Although the variance of \( r_t \) grows with \( t \), the tails of the distribution of \( r_t \) becomes longer and thinner. This makes the variance
grow, but because the tail events are rarely realized in-sample, $\mu$ can be chosen to make the process appear stationary.

Using the same simple model as above, it’s easy to see that the returns become

$$r_t = \mu + \epsilon_t \sigma_{t-1}$$

$$\sigma_t^2 = \sigma_{t-1}^2$$

Here, $E_{t-1}(r_t) = \mu$ and $V_{t-1}(r_t) = \sigma_{t-1}^2$. Thus, $V(r_t) = E(V_{t-1}(r_t)) + V(E_{t-1}(r_t)) = E(\sigma_{t-1}^2)$. Now $E_{t-1}(\sigma_t^2) = E_{t-1}(\sigma_{t-1}^2) = E_{t-1}(\mu^2 + \epsilon_t^2 \sigma_{t-1}^2 + 2 \mu \epsilon_t \sigma_{t-1}) = \mu^2 + \sigma_{t-1}^2$ and thus $E(\sigma_{t-1}^2) = \mu^2 + E(\sigma_{t-1}^2)$ which increases in $t$ because the mean $\mu$ is not subtracted from $r_t$ when calculating the variance estimate for the next period (this is how the MIDAS model has been used in practice).

Figure 2 shows the density of $r_2$, $r_5$, $r_{10}$, $r_{50}$, and $r_{100}$ when simulated with $\mu = 0.5$ (the densities are kernel estimates based on 1,000,000 simulated paths), as well as a typical sample path for $r_t$. The densities for $r_5$ and up are now virtually indistinguishable, even though the expected variance of $r_t$ is $E(\sigma_t) = 0.5^2(t-1) + 1$ (because $E(\sigma_{t+1}^2) = 0.5^2 + E(\sigma_t^2)$). Also, the sample path now doesn’t die out, but instead has a period of extreme outliers after approx. 5,400 steps.

Table 1 shows the theoretical expected variance, as well as the empirical variance of $r_t$ for different values of $t$. The empirical variances are based on 1,000,000 simulated paths. For small $t$ the empirical variance is close to the expected one, but when $t$ becomes larger the empirical variance is a lot smaller than the theoretical variance because the extreme outliers have not been realized in-sample.

Both of these specifications are undesirable: With $\mu = 0$ the distribution of $r_t$ has constant mean and variance, but otherwise it exhibits counterfactual features. With $\mu > 0$, the distribution of $r_t$ appears to have constant variance in simulations, but will occasionally blow up. Also, $\mu$ needs to be chosen carefully to obtain this apparent stationarity, and may not match a reasonable average return. For this reason, we propose a variation of the MIDAS model that circumvents these issues.
Figure 2: Density of $r_t$ when simulated from the MIDAS model with $\mu = 0.5$ in Section D.1.2.

Table 1: Theoretical and Empirical Variances.
The table shows the theoretical and empirical variance of $r_t$ for different values of $t$. The empirical variances are based on 1,000,000 simulated paths.

<table>
<thead>
<tr>
<th>$t$</th>
<th>True</th>
<th>Emp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>1.247</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>1.503</td>
</tr>
<tr>
<td>4</td>
<td>1.75</td>
<td>1.754</td>
</tr>
<tr>
<td>10</td>
<td>3.25</td>
<td>3.067</td>
</tr>
<tr>
<td>20</td>
<td>5.75</td>
<td>8.099</td>
</tr>
<tr>
<td>50</td>
<td>13.25</td>
<td>4.627</td>
</tr>
<tr>
<td>100</td>
<td>25.75</td>
<td>4.252</td>
</tr>
</tbody>
</table>

D.1.3 A Different Specification of Conditional Variance

Let the conditional variance estimate be given by

$$V_t^{\text{MIDAS}} = \omega + M_t \sum_{i=1}^{K} w_i(\theta) r^2_{t-i}$$

where $0 < \omega < 1$ and the weights $w_i$ now sum to a constant less than 1. To analyze this specification, consider again the simplest possible version:

$$r_t = \varepsilon_t \sigma_{t-1}$$
$$\sigma^2_t = \omega + \delta r^2_t = \omega + \delta \varepsilon^2_t \sigma^2_{t-1}$$
The general formula for the variance is found by induction. To get the idea for the induction assumption, write out the first few returns and variances: Let \( \sigma_0^2 = V_{0}^\text{MIDAS} \) be given. Then

\[
\begin{align*}
    r_1 &= \varepsilon_1 \sigma_0 \\
    \sigma_1^2 &= \omega + \delta r_1^2 = \omega + \delta \varepsilon_1^2 \sigma_0^2 \\
    r_2 &= \varepsilon_2 \sigma_1 = \varepsilon_2 \sqrt{\omega + \delta \varepsilon_1^2 \sigma_0^2} \\
    \sigma_2^2 &= \omega + \delta r_2^2 = \omega + \delta \varepsilon_2^2 + \delta^2 \sigma_0^2 \varepsilon_1^2 \varepsilon_2^2
\end{align*}
\]

Soon the pattern emerges:

\[
\sigma_t^2 = \omega \sum_{i=0}^{t-1} \delta^i \prod_{j=t+1-i}^{t} \varepsilon_j^2 + \sigma_0^2 \delta^i \prod_{i=1}^{t} \varepsilon_i^2.
\]

The induction step proceeds as follows. We have \( r_{t+1} = \varepsilon_{t+1} \sigma_t \) and then

\[
\begin{align*}
    \sigma_{t+1}^2 &= \omega + \delta r_{t+1}^2 \\
    &= \omega + \delta \varepsilon_{t+1} \sigma_t^2 \\
    &= \omega + \omega \delta \varepsilon_{t+1}^2 \sum_{i=0}^{t-1} \delta^i \prod_{j=t+1-i}^{t} \varepsilon_j^2 + \delta \varepsilon_{t+1} \sigma_t^2 \delta^i \prod_{i=1}^{t} \varepsilon_i^2 \\
    &= \omega + \omega \sum_{i=0}^{t-1} \delta^i \prod_{j=t+1-i}^{t} \varepsilon_j^2 + \sigma_0^2 \delta^{t+1} \prod_{i=1}^{t+1} \varepsilon_i^2 \\
    &= \omega + \omega \sum_{i=1}^{t} \delta^i \prod_{j=t+2-i}^{t} \varepsilon_j^2 + \sigma_0^2 \delta^{t+1} \prod_{i=1}^{t+1} \varepsilon_i^2
\end{align*}
\]

which completes the induction step. Now, assuming \( E(\sigma^2) \equiv E(\sigma_t^2) \) exists and is independent of \( t \), it must solve

\[
E(\sigma^2) = \omega + \delta E(\sigma^2)
\]

such that

\[
E(\sigma^2) = \frac{\omega}{1 - \delta}.
\]

Figure 3 shows the density of \( r_2, r_5, r_{10}, r_{50}, \) and \( r_{100} \) when simulating from the specification (the densities are kernel estimates based on 1,000,000 simulated paths), as well as a typical sample path for \( r_1 \). The densities for \( r_{10} \) and up are virtually indistinguishable, and the sample path shows clear evidence of volatility clustering without dying out or blowing up.

### D.2 The Simulation

I simulate from the model with \( \gamma = 0 \) as I’m primarily interested in the distribution of the parameter estimates under the null of \( \gamma = 0 \). In this case, the volatility of \( \varepsilon_t \) is time-varying but the mean of returns is constant. I choose \( \mu = 0.015 \) as this keeps the simulated paths roughly stationary (see the above discussion. The time-series is not stationary in theory, but in most samples it behaves as if it were). Simulations where the absolute value of a daily return is above 50% are discarded.

The weighting scheme used has exponential weights with a 2nd order polynomial, and the simulation experiments use the following parameter values:

\[
\mu = 0.015 \quad \gamma = 0 \quad \kappa_1 = -0.001 \quad \kappa_2 = -0.0001.
\]
Also, a lag length of 252 days is used, and all months are assumed to have 22 days (so $M_t = 22$ for all $t$). The resulting weights on lagged returns are shown in Figure 4.

The simulation proceeds as follows. For each path:

1. Simulate a burn-in period of 252 daily return with an annualized volatility of 15%.
2. Calculate the MIDAS estimate of conditional variance, $V_t^{MIDAS}$, based on the previous 252 returns.
3. Simulate 22 daily returns with mean $\frac{1}{22} (\mu + \gamma V_t^{MIDAS})$ and variance $\frac{1}{22} V_t^{MIDAS}$.
4. Move 22 days forward and return to step 2.
5. Continue until the required length of the time-series has been achieved.
6. Estimate the model based on the simulated data.

This is repeated 5,000 times.

### D.3 Large Sample Properties

This section considers a simulation experiment where the return process is simulated for 5000 months (approximately 416 years).

Figure 5 shows histograms of the estimates of the 4 parameters. The red lines denote true values, and the green lines denote sample averages. The estimates appear to be unbiased.

The parameter of primary interest is $\gamma$. This simulation experiment gives the empirical distribution of $\hat{\gamma}$ under the null of $\gamma = 0$. Table 2 shows percentiles of the empirical distribution of $\hat{\gamma}$. With the knowledge that in reality $\gamma$ should be positive, we could perform a one-sided test, in which case we would conclude that any value of $\gamma$ above 0.659 would be significantly different from zero at the 5%-level. Using a two-sided test,
we would conclude that any value of $\gamma$ above 0.831 would be significantly different from zero (recall that this is based on 5,000 months of observations).

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>0.5%</th>
<th>2.5%</th>
<th>5%</th>
<th>10%</th>
<th>90%</th>
<th>95%</th>
<th>97.5%</th>
<th>99.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>-1.5271</td>
<td>-1.0031</td>
<td>-0.7845</td>
<td>-0.5692</td>
<td>0.4661</td>
<td>0.6591</td>
<td>0.8309</td>
<td>1.2315</td>
</tr>
</tbody>
</table>

Table 2: Percentiles of the empirical distribution of $\hat{\gamma}$ when the model is simulated with $\gamma = 0$.

Figure 7 shows histograms of the $t$-statistics calculated using robust variance estimates for each of the four parameters. The $t$-stats using MLE variance estimates based on the Hessian are shown in Figure 8 and 9. Table 3 shows the variance of the estimates from the simulation, which can be viewed as the true variance of the estimates. Also, the table shows the mean of the robust variance estimates, the MLE Hessian estimate and the MLE score estimate. For $\gamma$ the robust variance estimate has the smallest bias.

I also calculate the rejection rate for the tests $\mu = \mu_0$, $\gamma = \gamma_0$, $\kappa_1 = \kappa_{10}$ and $\kappa_2 = \kappa_{20}$, where $\mu_0$, $\gamma_0$, $\kappa_{10}$ and $\kappa_{20}$ denote the true parameters used in the simulation. The test is done as a $t$-test, and the risks of incorrectly rejecting these hypothesis, a type 1 error, should be 5%. As shown in Table 4 when using robust standard error, the numbers are 5.74%, 5.10%, 6.20% and 10.52%. So for $\mu$, $\gamma$ and $\kappa_1$ a type 1 error occurs in approximately 5% of the simulations, but for $\kappa_2$ a type 1 error occurs more often. When using MLE standard errors, the rejection rates are a bit higher and thus farther from 5%.

To investigate how the precision of the estimates depends on the volatility of the simulated returns, Figure 10 first shows a histogram of the annualized volatility of the simulated return processes (calculated as $\text{std}(r) \cdot \sqrt{250}$). Most simulations have an annualized volatility between 10 and 20 percent, but for some simulations the volatility is higher.

Second, Figure 11 plots the estimate for a given simulation as a function of the annualized volatility for that simulation. The red lines show the true values of the parameters used in the simulations. Clearly, the
estimates of $\gamma$ becomes more precise when volatility is high. This is intuitive—if volatility is zero, $\gamma$ is not identified. It also looks like the estimates of $\mu, \kappa_1$ and $\kappa_2$ become more precise when volatility is high. Maybe we should look at the conditional distribution of $\hat{\gamma}$ given that the volatility of simulated returns is btw. something like 10% and 20%. This would make the empirical distribution less peaked, and make any given number less significant.

Figure 5: Parameter estimates for simulated data, when 5000 months of returns are simulated. The red lines denote true values, the green lines denote sample averages.

<table>
<thead>
<tr>
<th></th>
<th>True</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>2.34e-007</td>
<td>2.36e-007</td>
<td>2.29e-007</td>
<td>2.23e-007</td>
</tr>
<tr>
<td>Bias</td>
<td>2.49e-009</td>
<td>-4.76e-009</td>
<td>-1.08e-008</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>1.95e-001</td>
<td>1.90e-001</td>
<td>1.86e-001</td>
<td>1.83e-001</td>
</tr>
<tr>
<td>Bias</td>
<td>-4.19e-003</td>
<td>-8.19e-003</td>
<td>-1.11e-002</td>
<td></td>
</tr>
<tr>
<td>$\kappa_1$</td>
<td>3.21e-004</td>
<td>4.81e-004</td>
<td>2.93e-004</td>
<td>2.61e-004</td>
</tr>
<tr>
<td>Bias</td>
<td>1.60e-004</td>
<td>-2.83e-005</td>
<td>-6.04e-005</td>
<td></td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>2.31e-008</td>
<td>3.96e-008</td>
<td>2.14e-008</td>
<td>1.81e-008</td>
</tr>
<tr>
<td>Bias</td>
<td>1.65e-008</td>
<td>-1.64e-009</td>
<td>-5.01e-009</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Empirical variance of the simulated estimates (‘true variance’) and the mean of different variance estimates. The variance estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.
Figure 6: Scatter plot of the estimates of $\kappa_1$ and $\kappa_2$, showing that these estimates are strongly negatively correlated. The red lines show the true values of $\kappa_1$ and $\kappa_2$ used for the simulation.

Figure 7: Histogram of $t$-stats using Robust variance estimates for simulated data, when 5000 months of returns are simulated.
Figure 8: Histogram of $t$-stats using MLE variance estimates based on the Hessian for simulated data, when 5000 months of returns are simulated.

Figure 9: Histogram of $t$-stats using MLE variance estimates based on the score for simulated data, when 5000 months of returns are simulated.
<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu = \mu_0$</td>
<td>0.0574</td>
<td>0.0608</td>
<td>0.0646</td>
</tr>
<tr>
<td>$\gamma = \gamma_0$</td>
<td>0.0510</td>
<td>0.0542</td>
<td>0.0572</td>
</tr>
<tr>
<td>$\kappa_1 = \kappa_{10}$</td>
<td>0.0620</td>
<td>0.0706</td>
<td>0.0778</td>
</tr>
<tr>
<td>$\kappa_2 = \kappa_{20}$</td>
<td>0.1052</td>
<td>0.1050</td>
<td>0.1174</td>
</tr>
</tbody>
</table>

Table 4: Frequency of type 1 errors (rejection rate for the hypothesis that each parameter equals its true value), when 5000 months of returns are simulated. The standard errors of the estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.

Figure 10: Annualized volatility for each of the 5000 simulations, when 5000 months of returns are simulated.
Figure 11: Parameter estimates as a function of the annualized volatility of the simulated return process, when 5000 months of returns are simulated.
D.4 Small Sample Properties: 850 Months

This section considers a simulation experiment where the return process is simulated for 850 months (approximately the sample length available from CRSP). The simulation is repeated 5000 times. Figure 12 shows histograms of the estimates of the 4 parameters. The red lines denote true values, and the green lines denote sample averages. As before the estimates of $\mu$ and $\gamma$ appear to be unbiased. However, the estimates of $\kappa_1$ and $\kappa_2$ now have a small bias due to outliers.

![Histograms of Parameter Estimates](image)

Figure 12: Parameter estimates for simulated data, when 850 months of returns are simulated. The red lines denote true values, the green lines denote sample averages.

Table 5 shows percentiles of the empirical distribution of $\hat{\gamma}$. With the knowledge that in reality $\gamma$ should be positive, we could perform a one-sided test, in which case we would conclude that any value of $\gamma$ above 2.17 would be significantly different from zero. Using a two-sided test, we would conclude that any value of $\gamma$ above 2.80 would be significantly different from zero. Ghysels, Santa-Clara and Valkanov (2005) find $\gamma = 2.606$ with a $t$-stat of 6.7 using data from 1928 to 2000.

<table>
<thead>
<tr>
<th></th>
<th>0.5%</th>
<th>2.5%</th>
<th>5%</th>
<th>10%</th>
<th>90%</th>
<th>95%</th>
<th>97.5%</th>
<th>99.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>-5.6233</td>
<td>-3.7061</td>
<td>-2.8742</td>
<td>-2.0525</td>
<td>1.5130</td>
<td>2.1704</td>
<td>2.8015</td>
<td>4.2306</td>
</tr>
</tbody>
</table>

Table 5: Percentiles of the empirical distribution of $\hat{\gamma}$ when the model is simulated with $\gamma = 0$.

Figure 14 shows histograms of the $t$-stats using robust variance estimates for each of the four parameters. Histograms of the $t$-stats using MLE variance estimates based on the Hessian and score are shown in Figure 15 and 16.

Table 6 shows the variance of the estimates from the simulation, which can be viewed as the true variance of the estimates. Also, the table shows the mean of the robust variance estimates, the MLE Hessian estimate and the MLE score estimate. For $\gamma$ the robust variance estimate now has the largest bias.
Figure 13: Scatter plot of the estimates of $\kappa_1$ and $\kappa_2$, showing that these estimates are strongly negatively correlated. The red lines show the true values of $\kappa_1$ and $\kappa_2$ used for the simulation.

Figure 14: Histograms of $t$-stats using Robust variance estimates for simulated data, when 850 months of returns are simulated.
<table>
<thead>
<tr>
<th>True</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>3.97e-006</td>
<td>4.01e-006</td>
<td>3.72e-006</td>
</tr>
<tr>
<td>Bias</td>
<td>4.09e-008</td>
<td>-2.51e-007</td>
<td>-3.82e-007</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>2.44e+000</td>
<td>2.55e+000</td>
<td>2.45e+000</td>
</tr>
<tr>
<td>Bias</td>
<td>1.14e-001</td>
<td>1.67e-002</td>
<td>1.95e-002</td>
</tr>
<tr>
<td>$\kappa_1$</td>
<td>2.71e-002</td>
<td>4.74e-002</td>
<td>1.29e-002</td>
</tr>
<tr>
<td>Bias</td>
<td>2.04e-002</td>
<td>-1.41e-002</td>
<td>-1.78e-002</td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>1.90e-006</td>
<td>5.42e-006</td>
<td>1.19e-006</td>
</tr>
<tr>
<td>Bias</td>
<td>3.52e-006</td>
<td>-7.04e-007</td>
<td>-1.09e-006</td>
</tr>
</tbody>
</table>

Table 6: Empirical variance of the simulated estimates (‘true variance’) and the mean of different variance estimates, when simulating 850 months of returns. The variance estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.

I also calculate the rejection rate for the tests $\mu = \mu_0$, $\gamma = \gamma_0$, $\kappa_1 = \kappa_{10}$ and $\kappa_2 = \kappa_{20}$, where $\mu_0$, $\gamma_0$, $\kappa_{10}$ and $\kappa_{20}$ denote the true parameters used in the simulation. The test is done as a $t$-test, and the risks of incorrectly rejecting these hypothesis, a type 1 error, should be 5%. As shown in Table 7 the numbers are 5.32%, 4.84%, 7.34% and 11.54% using robust standard errors. So, as with 5000 months of simulated return, for $\mu$, $\gamma$ and $\kappa_1$ a type 1 error occurs in approximately 5% of the simulations, but for $\kappa_2$ a type 1 error occurs more often. The robust standard errors no longer perform the best across all parameters.

To investigate how the precision of the estimates depends on the volatility of the simulated returns, Figure 17 first shows the annualized volatility of the simulated return processes (calculated as $\text{std}(r) \cdot \sqrt{250}$). Most simulations have an annualized volatility between 10 and 20 percent, but for some simulations the volatility is higher.

Second, Figure 18 plots the estimate for a given simulation as a function of the annualized volatility for

![Histograms of t-stats](image-url)

Figure 15: Histograms of $t$-stats using MLE variance estimates based on the Hessian for simulated data, when 850 months of returns are simulated.
that simulation. The red lines show the true values of the parameters used in the simulations. Clearly, the estimate of $\gamma$ becomes more precise when volatility is high. It also looks like the estimates of $\mu, \kappa_1$ and $\kappa_2$ become more precise when volatility is high. **Maybe we should look at the conditional distribution of $\gamma$ given that the volatility of simulated returns is btw. something like 10% and 20%**.
Figure 17: Annualized volatility for each of the 850 simulations, when 850 months of returns are simulated.

Figure 18: Parameter estimates as a function of the annualized volatility of the simulated return process, when 850 months of returns are simulated.
Finally, this section considers a simulation experiment where the return process is simulated for 450 months (approximately half of the period available from CRSP). Figure 19 shows histograms of the estimates of the 4 parameters. The red lines denote true values, and the green lines denote sample averages. As before, the estimates of $\mu$ and $\gamma$ appear to be unbiased. However, the estimates of $\kappa_1$ and $\kappa_2$ now have some bias due to outliers.

Table 8 shows percentiles of the empirical distribution of $\hat{\gamma}$. With the knowledge that in reality $\gamma$ should be positive, we could perform a one-sided test, in which case we would conclude that any value of $\gamma$ above 3.42 would be significantly different from zero. Using a two-sided test, we would conclude that any value of $\gamma$ above 4.56 would be significantly different from zero. Ghysels, Santa-Clara and Valkanov (2005) find $\gamma = 1.547$ with a $t$-stat of 3.38 using data from 1928 to 1963, and $\gamma = 3.748$ with a $t$-stat of 8.61 using data from 1964 to 2000.

<table>
<thead>
<tr>
<th>0.5%</th>
<th>2.5%</th>
<th>5%</th>
<th>10%</th>
<th>90%</th>
<th>95%</th>
<th>97.5%</th>
<th>99.5%</th>
</tr>
</thead>
</table>

Table 8: Percentiles of the empirical distribution of $\hat{\gamma}$ when the model is simulated with $\gamma = 0$.

Figure 20 shows histograms of the $t$-stats using robust variance estimates for each of the four parameters. The $t$-stats using MLE variance estimates based on the Hessian and score are shown in Figure ?? and ??.

Table ?? shows the variance of the estimates from the simulation, which can be viewed as the true variance of the estimates. Also, the table shows the mean of the robust variance estimates, the MLE Hessian estimate and the MLE score estimate. As with 850 months of observations, the robust variance estimate has the
Figure 20: Scatter plot of the estimates of $\kappa_1$ and $\kappa_2$, showing that these estimates are strongly negatively correlated. The red lines show the true values of $\kappa_1$ and $\kappa_2$ used for the simulation.

Figure 21: Histograms of $t$-stats using robust variance estimates for simulated data, when 450 months of returns are simulated.
I also calculate the rejection rate for the tests $\mu = \mu_0$, $\gamma = \gamma_0$, $\kappa_1 = \kappa_{10}$ and $\kappa_2 = \kappa_{20}$, where $\mu_0$, $\gamma_0$, $\kappa_{10}$ and $\kappa_{20}$ denote the true parameters used in the simulation. The test is done as a $t$-test, and the risks of
Table 9: Empirical variance of the simulated estimates (‘true variance’) and the mean of different variance estimates, when simulating 450 months of returns. The variance estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>True</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>1.38e-005</td>
<td>1.38e-005</td>
<td>1.21e-005</td>
<td>1.15e-005</td>
</tr>
<tr>
<td>Bias</td>
<td>-7.48e-008</td>
<td>-1.78e-006</td>
<td>-2.30e-006</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>6.09e+000</td>
<td>6.44e+000</td>
<td>5.92e+000</td>
<td>5.98e+000</td>
</tr>
<tr>
<td>Bias</td>
<td>3.55e-001</td>
<td>-1.63e-001</td>
<td>-1.09e-001</td>
<td></td>
</tr>
<tr>
<td>$\kappa_1$</td>
<td>3.75e-001</td>
<td>7.46e-001</td>
<td>1.82e-001</td>
<td>1.38e-001</td>
</tr>
<tr>
<td>Bias</td>
<td>3.71e-001</td>
<td>-1.93e-001</td>
<td>-2.37e-001</td>
<td></td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>1.93e-005</td>
<td>2.83e-005</td>
<td>1.01e-005</td>
<td>7.82e-006</td>
</tr>
<tr>
<td>Bias</td>
<td>8.95e-006</td>
<td>-9.23e-006</td>
<td>-1.15e-005</td>
<td></td>
</tr>
</tbody>
</table>

incorrectly rejecting these hypothesis, a type 1 error, should be 5%. As shown in Table 10 the numbers are 5.16%, 5.52%, 9.40% and 12.58% using robust standard errors. As before, for $\mu$ and $\gamma$ a type 1 error occurs in approximately 5% of the simulations, but for both $\kappa_1$ and $\kappa_2$ a type 1 error occurs more often.

Table 10: Frequency of type 1 errors (rejection rate for the hypothesis that each parameter equals its true value), when 450 months of returns are simulated. The standard errors of the estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu = \mu_0$</td>
<td>0.0516</td>
<td>0.0606</td>
<td>0.0656</td>
</tr>
<tr>
<td>$\gamma = \gamma_0$</td>
<td>0.0552</td>
<td>0.0580</td>
<td>0.0590</td>
</tr>
<tr>
<td>$\kappa_1 = \kappa_{10}$</td>
<td>0.0940</td>
<td>0.0548</td>
<td>0.0590</td>
</tr>
<tr>
<td>$\kappa_2 = \kappa_{20}$</td>
<td>0.1258</td>
<td>0.0930</td>
<td>0.0944</td>
</tr>
</tbody>
</table>

To investigate how the precision of the estimates depends on the volatility of the simulated returns, Figure 24 first shows the annualized volatility of the simulated return processes (calculated as $\text{std}(r) \cdot \sqrt{250}$). Most simulations have an annualized volatility between 10 and 20 percent, but for some simulations the volatility is higher, sometimes as high as 80% annualized.

Second, Figure 25 plots the estimate for a given simulation as a function of the annualized volatility for that simulation. The red lines show the true values of the parameters used in the simulations. Clearly, the estimate of $\gamma$ becomes more precise when volatility is high, and this is now also clearly the case for the estimates of $\kappa_1$ and $\kappa_2$. Maybe we should look at the conditional distribution of $\hat{\gamma}$ given that the volatility of simulated returns is btw. something like 10% and 20%.
Figure 24: Annualized volatility for each of the 450 simulations, when 450 months of returns are simulated.

Figure 25: Parameter estimates as a function of the annualized volatility of the simulated return process, when 450 months of returns are simulated.
D.6 Large Sample Properties: Alternative Specification

I now consider the alternative specification of variance, in which

\[ V_{t}^{\text{MIDAS}} = \omega + \delta \sum_{i=0}^{K} w_{i}(\theta)r_{t-i}^{2}. \]  

(83)

The unconditional variance is given by \( \omega/(1-\delta) \). Hence, I simulate the model by specifying the unconditional volatility to be 15% annualized and let the variance be given by

\[ V_{t}^{\text{MIDAS}} = \text{Var}(r)(1-\delta) + \delta \sum_{i=0}^{K} w_{i}(\theta)r_{t-i}^{2}. \]  

(84)

Next, I also estimate the model using this specification, which only requires estimating \( \delta \) and not \( \omega \) (this is sometimes known as 'volatility targeting' and is also common when estimating GARCH models). Because \( \delta \) is required to be in \((0, 1)\) to ensure a positive variance, I use the transformation \( \delta = \exp(\phi)/(1 + \exp(\phi)) \) where \( \phi \in (-\infty, \infty) \). For the simulations I choose \( \phi = 1 \) which corresponds to \( \delta = 0.7311 \).

When simulating using this specification, the return process no longer blows up or dies out. Hence, I now take \( \mu = 0 \), but keep the other parameters as before. Thus,

\[ \mu = 0.0 \quad \gamma = 0.0 \quad \kappa_{1} = -0.001 \quad \kappa_{2} = -0.0001 \quad \phi = 1 \]  

(85)

This section considers a simulation experiment where the return process is simulated for 5000 months (approximately 416 years).

Figure 26 shows histograms of the estimates of the 4 parameters. The red lines denote true values, and the green lines denote sample averages. The figure now includes a histogram of the estimates of \( \phi \). All estimates appear to be unbiased.

The parameter of primary interest is \( \gamma \). This simulation experiment gives the empirical distribution of \( \hat{\gamma} \) under the null of \( \gamma = 0 \). Table 11 shows percentiles of the empirical distribution of \( \hat{\gamma} \). With the knowledge that in reality \( \gamma \) should be positive, we could perform a one-sided test, in which case we would conclude that any value of \( \gamma \) above 1.15 would be significantly different from zero at the 5%-level. Using a two-sided test, we would conclude that any value of \( \gamma \) above 1.38 would be significantly different from zero (recall that this is based on 5,000 months of observations). Note that the distribution of \( \gamma \) found for this specification of the MIDAS model is much wider than before. Think about why: is it because volatility is lower on average, or is it because vol of vol is different?

<table>
<thead>
<tr>
<th>0.5%</th>
<th>2.5%</th>
<th>5%</th>
<th>10%</th>
<th>90%</th>
<th>95%</th>
<th>97.5%</th>
<th>99.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma )</td>
<td>-1.9553</td>
<td>-1.4784</td>
<td>-1.2061</td>
<td>-0.9278</td>
<td>0.9046</td>
<td>1.1532</td>
<td>1.3804</td>
</tr>
</tbody>
</table>

Table 11: Percentiles of the empirical distribution of \( \hat{\gamma} \) when the model is simulated with \( \gamma = 0 \).

Figure 27 shows histograms of the t-stats using robust variance estimates for each of the four parameters, and the t-stats using MLE variance estimates based on the Hessian and score are shown in Figure 28 and 29.

Table 12 shows the variance of the estimates from the simulation, which can be viewed as the true variance of the estimates. Also, the table shows the mean of the robust variance estimates, the MLE Hessian estimate and the MLE score estimate. The size of the bias is an order of magnitude smaller than the mean.

I also calculate the rejection rate for the tests \( \mu = \mu_{0}, \gamma = \gamma_{0}, \kappa_{2} = \kappa_{20} \) and \( \phi = \phi_{0} \), where \( \mu_{0}, \gamma_{0}, \kappa_{20}, \) and \( \phi_{0} \) denote the true parameters used in the simulation. The test is done as a t-test, and the risks of incorrectly rejecting these hypothesis, a type 1 error, should be 5%. As shown in Table 13 when using robust standard error, the numbers are 4.86%, 4.78%, 4.76%, 6.88% and 3.66%.

Figure 30 shows the annualized volatility of the simulated return processes (calculated as \( \text{std}(r) \cdot \sqrt{250} \)). With this specification of the conditional variance, the simulated variance is always close to the target of 15%.
Figure 26: Parameter estimates for simulated data, when 5000 months of returns are simulated. The red lines denote true values, the green lines denote sample averages.
Figure 27: Histograms of t-stats using robust variance estimates for simulated data, when 5000 months of returns are simulated.

<table>
<thead>
<tr>
<th></th>
<th>True</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>7.88e-006</td>
<td>7.96e-006</td>
<td>7.82e-006</td>
<td>7.84e-006</td>
</tr>
<tr>
<td>Bias</td>
<td>8.57e-008</td>
<td>-5.28e-008</td>
<td>-3.13e-008</td>
<td>-3.13e-008</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>2.14e+000</td>
<td>2.17e+000</td>
<td>2.13e+000</td>
<td>2.14e+000</td>
</tr>
<tr>
<td>Bias</td>
<td>2.62e-002</td>
<td>-1.14e-002</td>
<td>-5.24e-003</td>
<td>-5.24e-003</td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>3.82e+000</td>
<td>4.40e+000</td>
<td>3.87e+000</td>
<td>3.75e+000</td>
</tr>
<tr>
<td>Bias</td>
<td>5.79e-001</td>
<td>5.35e-002</td>
<td>-7.11e-002</td>
<td>-7.11e-002</td>
</tr>
<tr>
<td>$\phi$</td>
<td>1.38e-001</td>
<td>1.43e-001</td>
<td>1.39e-001</td>
<td>1.38e-001</td>
</tr>
<tr>
<td>Bias</td>
<td>4.59e-003</td>
<td>4.80e-004</td>
<td>7.04e-005</td>
<td>7.04e-005</td>
</tr>
</tbody>
</table>

Table 12: Empirical variance of the simulated estimates (‘true variance’) and the mean of different variance estimates. The variance estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.
Figure 28: Histograms of $t$-stats using MLE variance estimates based on the Hessian for simulated data, when 5000 months of returns are simulated.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu = \mu_0$</td>
<td>0.0478</td>
<td>0.0488</td>
<td>0.0482</td>
</tr>
<tr>
<td>$\gamma = \gamma_0$</td>
<td>0.0476</td>
<td>0.0486</td>
<td>0.0480</td>
</tr>
<tr>
<td>$\kappa_2 = \kappa_{20}$</td>
<td>0.0688</td>
<td>0.0652</td>
<td>0.0696</td>
</tr>
<tr>
<td>$\phi = \phi_0$</td>
<td>0.0366</td>
<td>0.0382</td>
<td>0.0392</td>
</tr>
</tbody>
</table>

Table 13: Frequency of type 1 errors (rejection rate for the hypothesis that each parameter equals its true value), when 5000 months of returns are simulated. The standard errors of the estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.
Figure 29: Histograms of $t$-stats using MLE variance estimates based on the score for simulated data, when 5000 months of returns are simulated.
Figure 30: Annualized volatility for each of the 5000 simulations, when 5000 months of returns are simulated.
D.7 Small Sample Properties, 850 Months: Alternative Specification

This section considers a simulation experiment where the return process is simulated for 850 months (approximately the sample length available from CRSP). The simulation is repeated 5000 times. Figure 31 shows histograms of the estimates of the 5 parameters as well as the annualized volatility of the simulated returns. The red lines denote true values, and the green lines denote sample averages. As before the estimates of $\mu$ and $\gamma$ appear to be unbiased. Because of the transformation $\phi = \exp(\tilde{\phi})/(1 + \exp(\tilde{\phi}))$, the estimate of $\tilde{\phi}$ now has a cluster between 10 and 15, which corresponds to $\phi \approx 1$. This is, the optimizer would really like to drive $\phi$ above 1, but because of the restriction $0 < \phi < 1$ the estimates instead cluster around 1, corresponding to large values of $\tilde{\phi}$ between 10 and 15.

Table 14 shows percentiles of the empirical distribution of $\hat{\gamma}$, which is now much wider than before. With the knowledge that in reality $\gamma$ should be positive, we could perform a one-sided test, in which case we would conclude that any value of $\gamma$ above 6.59 would be significantly different from zero. Using a two-sided test, we would conclude that any value of $\gamma$ above 7.95 would be significantly different from zero.

<table>
<thead>
<tr>
<th>0.5%</th>
<th>2.5%</th>
<th>5%</th>
<th>10%</th>
<th>90%</th>
<th>95%</th>
<th>97.5%</th>
<th>99.5%</th>
</tr>
</thead>
</table>

Table 14: Percentiles of the empirical distribution of $\hat{\gamma}$ when the model is simulated with $\gamma = 0$.

Figure 32 shows histograms of the $t$-stats using robust variance estimates for each of the four parameters, and the $t$-stats using MLE variance estimates based on the Hessian and score are shown in Figure 33 and 34.

Table 15 shows the variance of the estimates from the simulation, which can be viewed as the true variance of the estimates. Also, the table shows the mean of the robust variance estimates, the MLE Hessian estimate and the MLE score estimate. For $\mu$ and $\gamma$ the bias of the variance estimates is an order of magnitude smaller than the variance itself. However, for $\kappa_2$ and $\phi$ the bias in the variance is significant.

<table>
<thead>
<tr>
<th></th>
<th>True</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>6.46e-005</td>
<td>8.92e-005</td>
<td>6.79e-005</td>
<td>6.32e-005</td>
</tr>
<tr>
<td>Bias</td>
<td>2.46e-005</td>
<td>3.29e-006</td>
<td>-1.40e-006</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>1.76e+001</td>
<td>2.38e+001</td>
<td>1.83e+001</td>
<td>1.72e+001</td>
</tr>
<tr>
<td>Bias</td>
<td>6.24e+000</td>
<td>7.23e-001</td>
<td>-4.54e-001</td>
<td></td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>9.81e+001</td>
<td>1.62e+002</td>
<td>7.69e+001</td>
<td>8.73e+001</td>
</tr>
<tr>
<td>Bias</td>
<td>6.42e+001</td>
<td>-2.12e+001</td>
<td>-1.08e+001</td>
<td></td>
</tr>
<tr>
<td>$\phi$</td>
<td>6.53e+000</td>
<td>4.74e+001</td>
<td>5.52e+003</td>
<td>6.22e+008</td>
</tr>
<tr>
<td>Bias</td>
<td>4.08e+001</td>
<td>5.51e+003</td>
<td>6.22e+008</td>
<td></td>
</tr>
</tbody>
</table>

Table 15: Empirical variance of the simulated estimates (‘true variance’) and the mean of different variance estimates, when simulating 850 months of returns. The variance estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.

I also calculate the rejection rate for the tests $H_0: \mu = \mu_0, \gamma = \gamma_0, \kappa_2 = \kappa_{20}$, and $H_0: \phi = \phi_0$ where $\mu_0, \gamma_0, \kappa_{20}$, and $\phi_0$ denote the true parameters used in the simulation. The test is done as a $t$-test, and the risks of incorrectly rejecting these hypothesis, a type 1 error, should be 5%. As shown in Table 16 the numbers are 3.24%, 3.02%, 15.14% and 7.44% using robust standard errors. So, as with 5000 months of simulated return, for $\mu, \gamma$ and $\kappa_2$ a type 1 error occurs in approximately 5% of the simulations, but for $\kappa_2$ a type 1 error occurs more often.

Figure 35 shows the annualized volatility of the simulated return processes (calculated as $\text{std}(r) \cdot \sqrt{250}$).
Figure 31: Parameter estimates for simulated data, when 850 months of returns are simulated. The red lines denote true values, the green lines denote sample averages.
Figure 32: Histograms of t-stats using robust variance estimates for simulated data, when 850 months of returns are simulated.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu = \mu_0$</td>
<td>0.0324</td>
<td>0.0334</td>
<td>0.0370</td>
</tr>
<tr>
<td>$\gamma = \gamma_0$</td>
<td>0.0302</td>
<td>0.0326</td>
<td>0.0352</td>
</tr>
<tr>
<td>$\kappa_2 = \kappa_{20}$</td>
<td>0.1514</td>
<td>0.1362</td>
<td>0.1292</td>
</tr>
<tr>
<td>$\phi = \phi_0$</td>
<td>0.0744</td>
<td>0.0358</td>
<td>0.0364</td>
</tr>
</tbody>
</table>

Table 16: Frequency of type 1 errors (rejection rate for the hypothesis that each parameter equals its true value), when 850 months of returns are simulated. The standard errors of the estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.
Figure 33: Histograms of t-stats using MLE variance estimates based on the Hessian for simulated data, when 850 months of returns are simulated.
Figure 34: Histograms of $t$-stats using MLE variance estimates based on the score for simulated data, when 850 months of returns are simulated.
Figure 35: Annualized volatility for each of the 5000 simulations, when 850 months of returns are simulated.
D.8 Small Sample Properties, 450 Months: Alternative Specification

Finally, this section considers a simulation experiment where the return process is simulated for 450 months (approximately half of the period available from CRSP). Figure 36 shows histograms of the estimates of the 4 parameters. The red lines denote true values, and the green lines denote sample averages. As before the estimates of $\mu$ and $\gamma$ appear to be unbiased. However, the estimates of $\kappa^2$ now have some bias due to outliers, and the estimate of $\phi$ is heavily biased due to the clustering of estimates around 10-15 (corresponding to $\delta \approx 1$). For both $\mu, \gamma$ and $\kappa^2$ we also see some clustering of estimates close to the boundaries I've imposed when doing the optimization.

![Parameter estimates for simulated data, when 450 months of returns are simulated. The red lines denote true values, the green lines denote sample averages.](image-url)

Figure 36: Parameter estimates for simulated data, when 450 months of returns are simulated. The red lines denote true values, the green lines denote sample averages.
Table 17 shows percentiles of the empirical distribution of $\hat{\gamma}$, which is now very wide. With the knowledge that in reality $\gamma$ should be positive, we could perform a one-sided test, in which case we would conclude that any value of $\gamma$ above 9.85 would be significantly different from zero. Using a two-sided test, we would conclude that any value of $\gamma$ above 13.13 would be significantly different from zero.

<table>
<thead>
<tr>
<th>0.5%</th>
<th>2.5%</th>
<th>5%</th>
<th>10%</th>
<th>90%</th>
<th>95%</th>
<th>97.5%</th>
<th>99.5%</th>
</tr>
</thead>
</table>

Table 17: Percentiles of the empirical distribution of $\hat{\gamma}$ when the model is simulated with $\gamma = 0$.

Figure 37 shows histograms of the $t$-stats using robust variance estimates for each of the four parameters, and the $t$-stats using MLE variance estimates based on the Hessian and score are shown in Figure 38 and 39.

Table 18 shows the variance of the estimates from the simulation, which can be viewed as the true variance of the estimates. Also, the table shows the mean of the robust variance estimates, the MLE Hessian estimate and the MLE score estimate. There is now a heavy bias in all variance estimates due to outliers!

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>True</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>2.37e-004</td>
<td>8.99e-002</td>
<td>-2.70e-001</td>
<td>4.43e+005</td>
</tr>
<tr>
<td>Bias</td>
<td>8.96e-002</td>
<td>-2.71e-001</td>
<td>4.43e+005</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>6.47e+001</td>
<td>2.45e+004</td>
<td>-7.79e+004</td>
<td>1.17e+011</td>
</tr>
<tr>
<td>Bias</td>
<td>2.45e+004</td>
<td>-7.80e+004</td>
<td>1.17e+011</td>
<td></td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>4.21e+002</td>
<td>5.29e+003</td>
<td>-3.63e+006</td>
<td>1.63e+013</td>
</tr>
<tr>
<td>Bias</td>
<td>4.87e+003</td>
<td>-3.63e+006</td>
<td>1.63e+013</td>
<td></td>
</tr>
<tr>
<td>$\phi$</td>
<td>1.63e+001</td>
<td>1.31e+002</td>
<td>2.40e+004</td>
<td>3.58e+010</td>
</tr>
<tr>
<td>Bias</td>
<td>1.14e+002</td>
<td>2.40e+004</td>
<td>3.58e+010</td>
<td></td>
</tr>
</tbody>
</table>

Table 18: Empirical variance of the simulated estimates ('true variance') and the mean of different variance estimates, when simulating 450 months of returns. The variance estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.

I also calculate the rejection rate for the tests $\mu = \mu_0$, $\gamma = \gamma_0$, $\kappa_2 = \kappa_{20}$, and $\phi = \phi_0$, where $\mu_0$, $\gamma_0$, $\kappa_{20}$, and $\phi_0$ denote the true parameters used in the simulation. The test is done as a $t$-test, and the risks of incorrectly rejecting these hypothesis, a type 1 error, should be 5%. As shown in Table 19 the numbers are 2.54%, 2.44%, 18.70% and 13.78% using robust standard errors. For $\mu$ and $\gamma$ a type 1 error occurs in approximately 2.5% the simulations—half of the target frequency, but for $\kappa_2$ a type 1 error occurs more often, and for $\phi$ it depends on the variance estimate used.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Robust</th>
<th>MLE Hessian</th>
<th>MLE Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu = \mu_0$</td>
<td>0.0254</td>
<td>0.0242</td>
<td>0.0212</td>
</tr>
<tr>
<td>$\gamma = \gamma_0$</td>
<td>0.0244</td>
<td>0.0224</td>
<td>0.0204</td>
</tr>
<tr>
<td>$\kappa_2 = \kappa_{20}$</td>
<td>0.1870</td>
<td>0.1552</td>
<td>0.1346</td>
</tr>
<tr>
<td>$\phi = \phi_0$</td>
<td>0.1378</td>
<td>0.0352</td>
<td>0.0316</td>
</tr>
</tbody>
</table>

Table 19: Frequency of type 1 errors (rejection rate for the hypothesis that each parameter equals its true value), when 450 months of returns are simulated. The standard errors of the estimates are calculated in three ways: Using White’s method (robust); using MLE standard errors based on the Hessian; and using MLE standard errors based on the score.

Figure 40 shows the annualized volatility of the simulated return processes (calculated as $std(r) \cdot \sqrt{250}$).
Figure 37: Histograms of $t$-stats using robust variance estimates for simulated data, when 450 months of returns are simulated.

References

Figure 38: Histograms of $t$-stats using MLE variance estimates based on the Hessian for simulated data, when 450 months of returns are simulated.
Figure 39: Histograms of $t$-stats using MLE variance estimates based on the score for simulated data, when 450 months of returns are simulated.
Figure 40: Annualized volatility for each of the 5000 simulations, when 450 months of returns are simulated.