MIDAS in gretl

Allin Cottrell

Wake Forest University

Gretl conference, Athens 2017

Subject matter

"Mixed Data Sampling" in gretl

See http://gretl.sourceforge.net/midas/, in particular

- midas_gretl.pdf : newly revised guide
- midas-supp.pdf : supplement with forecasting experiments, etc.

I will talk about some of the points in each of these documents.

Mixed frequency data

How to combine frequencies in a single data file/dataset? "Spread" the higher-frequency data.

Here's a slice of MIDAS data...

	gdpc96	indpro_m3	indpro_m2	indpro_m1
1947:1	1934.47	14.3650	14.2811	14.1973
1947:2	1932.28	14.3091	14.3091	14.2532
1947:3	1930.31	14.4209	14.3091	14.2253
1947:4	1960.70	14.8121	14.7562	14.5606
1948:1	1989.54	14.7563	14.9240	14.8960
1948:2	2021.85	15.2313	15.0357	14.7842

Creating a MIDAS dataset

Importation from a database is easy (script):

```
clear
open fedstl.bin
data gdpc96
data indpro --compact=spread
store gdp_indpro.gdt
```

Other methods:

- Create two datasets, compact the high-frequency one, then use append.
- Use matrices.
- Use `join`.

Creating a MIDAS dataset

Importation from a database is easy (script):

```
clear
open fedstl.bin
data gdpc96
data indpro --compact=spread
store gdp_indpro.gdt
```

Other methods:

- Create two datasets, compact the high-frequency one, then use append.
- Use matrices.
- Use `join`.

MIDAS lists

A MIDAS list is a list of *m* series holding per-period values of a single high-frequency series, arranged in the order of most recent first.

E.g.

```
list INDPRO = indpro_m3 indpro_m2 indpro_m1
```

Or (if the series are in the right order)

```
list INDPRO = indpro_m*
```

May also want to do

```
setinfo INDPRO --midas
```

High frequency lags

The regular `lags` function works on the base frequency of the dataset.

But we have the dedicated function `hflags`:

```
list INDPRO = indpro_m*
setinfo INDPRO --midas
# create high-frequency lags 1 to 6
list IPL = hflags(1, 6, INDPRO)
list IPL print
```

- The length of the list argument determines the "compaction factor", *m*.
- Lags are specified in high-frequency terms.
- Ordering of the generated series by lag is automatic.

[back to gretl for a look]

High frequency lags

The regular `lags` function works on the base frequency of the dataset.

But we have the dedicated function `hflags`:

```
list INDPRO = indpro_m*
setinfo INDPRO --midas
# create high-frequency lags 1 to 6
list IPL = hflags(1, 6, INDPRO)
list IPL print
```

- The length of the list argument determines the "compaction factor", *m*.
- Lags are specified in high-frequency terms.
- Ordering of the generated series by lag is automatic.

[back to gretl for a look]

High frequency lags

The regular `lags` function works on the base frequency of the dataset.

But we have the dedicated function `hflags`:

```
list INDPRO = indpro_m*
setinfo INDPRO --midas
# create high-frequency lags 1 to 6
list IPL = hflags(1, 6, INDPRO)
list IPL print
```

- The length of the list argument determines the "compaction factor", *m*.
- Lags are specified in high-frequency terms.
- Ordering of the generated series by lag is automatic.

[back to gretl for a look]

High frequency lags

The regular `lags` function works on the base frequency of the dataset.

But we have the dedicated function `hflags`:

```
list INDPRO = indpro_m*
setinfo INDPRO --midas
# create high-frequency lags 1 to 6
list IPL = hflags(1, 6, INDPRO)
list IPL print
```

- The length of the list argument determines the "compaction factor", *m*.
- Lags are specified in high-frequency terms.
- Ordering of the generated series by lag is automatic.

[back to gretl for a look]

Leads and nowcasting

Where is high-frequency (HF) lag zero?

Unlike the single-frequency case, it's a matter of convention.

First, let's just take a look at the time-line, using the quarterly plus monthly case—and getting used to right-to-left time!

Now, if we've got that. . .

Leads and nowcasting

Where is high-frequency (HF) lag zero?

Unlike the single-frequency case, it's a matter of convention.

First, let's just take a look at the time-line, using the quarterly plus monthly case—and getting used to right-to-left time!

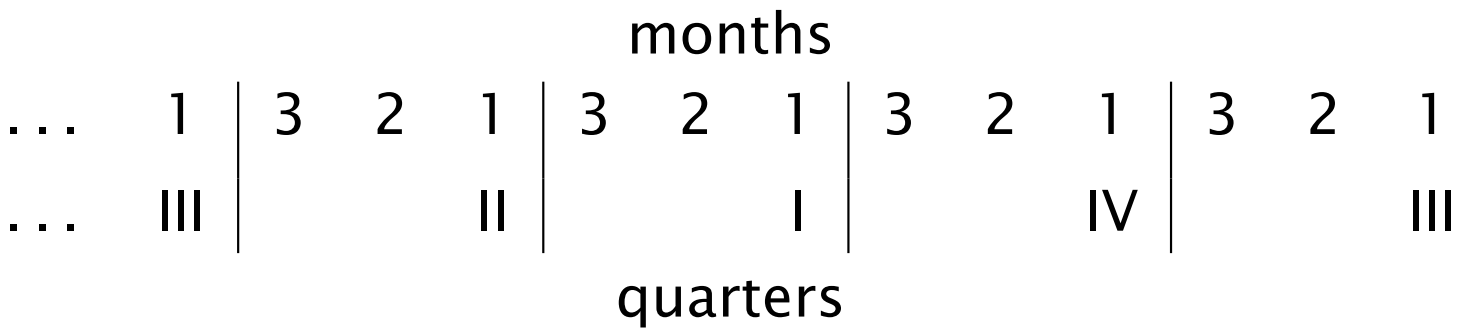
Now, if we've got that. . .

Leads and nowcasting

Where is high-frequency (HF) lag zero?

Unlike the single-frequency case, it's a matter of convention.

First, let's just take a look at the time-line, using the quarterly plus monthly case—and getting used to right-to-left time!



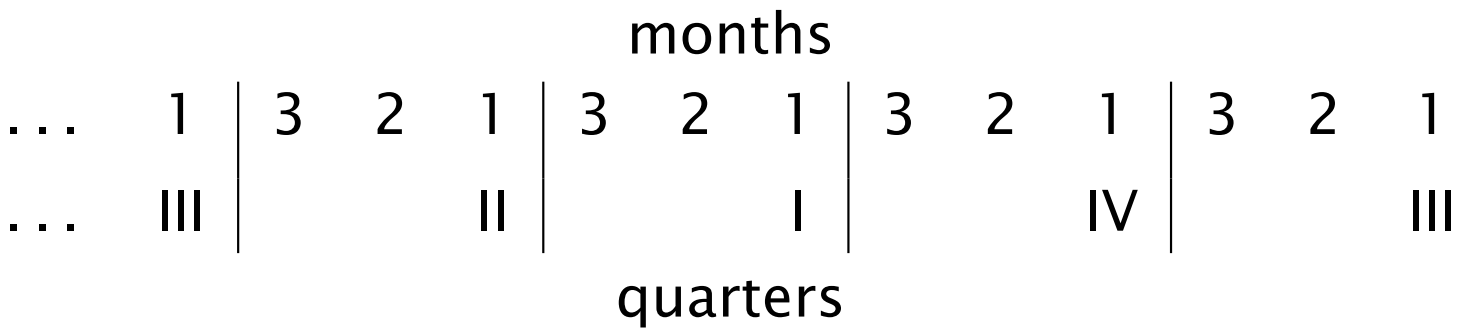
Now, if we've got that. . .

Leads and nowcasting

Where is high-frequency (HF) lag zero?

Unlike the single-frequency case, it's a matter of convention.

First, let's just take a look at the time-line, using the quarterly plus monthly case—and getting used to right-to-left time!



Now, if we've got that. . .

Competing conventions

For MIDAS Matlab Toolbox and gretl:

3	2	1	3	2	1	3	2	1	3	2	1
		...	−2	−1	0	1	2	3	...		
		II			I			IV			III

But for R (package midasr):

3	2	1	3	2	1	3	2	1	3	2	1
		...	0	1	2	3	4	5	...		
		II			I			IV			III

(It took me a while to figure this out!)

Competing conventions

For MIDAS Matlab Toolbox and gretl:

3	2	1	3	2	1	3	2	1	3	2	1
		...	−2	−1	0	1	2	3	...		
		II			I			IV			III

But for R (package midasr):

3	2	1	3	2	1	3	2	1	3	2	1
		...	0	1	2	3	4	5	...		
		II			I			IV			III

(It took me a while to figure this out!)

High frequency differences

The regular functions `diff` and `ldiff` will not do what you (probably) want...

But we have the dedicated functions `hfdiff` and `hfldiff`.

```
list INDPRO = indpro_m*
setinfo INDPRO --midas
list dX = hfldiff(INDPRO, 100)
```

The last argument is an optional multiplier, applied to all generated series.

Then, probably,

```
list dXL = hflags(1, 10, dX)
```

Or you can nest the two functions:

```
list dXL = hflags(1, 10, hfldiff(X, 100))
```

High frequency differences

The regular functions `diff` and `ldiff` will not do what you (probably) want...

But we have the dedicated functions `hfdiff` and `hfldiff`.

```
list INDPRO = indpro_m*
setinfo INDPRO --midas
list dX = hfldiff(INDPRO, 100)
```

The last argument is an optional multiplier, applied to all generated series.

Then, probably,

```
list dXL = hflags(1, 10, dX)
```

Or you can nest the two functions:

```
list dXL = hflags(1, 10, hfldiff(X, 100))
```

High frequency differences

The regular functions `diff` and `ldiff` will not do what you (probably) want...

But we have the dedicated functions `hfdiff` and `hfldiff`.

```
list INDPRO = indpro_m*
setinfo INDPRO --midas
list dX = hfldiff(INDPRO, 100)
```

The last argument is an optional multiplier, applied to all generated series.

Then, probably,

```
list dXL = hflags(1, 10, dX)
```

Or you can nest the two functions:

```
list dXL = hflags(1, 10, hfldiff(X, 100))
```

High frequency differences

The regular functions `diff` and `ldiff` will not do what you (probably) want...

But we have the dedicated functions `hfdiff` and `hfldiff`.

```
list INDPRO = indpro_m*
setinfo INDPRO --midas
list dX = hfldiff(INDPRO, 100)
```

The last argument is an optional multiplier, applied to all generated series.

Then, probably,

```
list dXL = hflags(1, 10, dX)
```

Or you can nest the two functions:

```
list dXL = hflags(1, 10, hfldiff(X, 100))
```

Parsimonious parameterizations

Simplest parameterization is "unrestricted MIDAS" (U-MIDAS); can be estimated by OLS. E.g.

$$y_t = \alpha + \beta y_{t-1} + \sum_{i=1}^{p} \gamma_i x_{\tau-i}$$

(where τ indicates "high-frequency time").

But more common to use something more parsimonious, for example:

- Normalized exponential Almon
- Normalized beta distribution (3 variants)
- (non-normalized) Almon polynomial

Parsimonious parameterizations

Simplest parameterization is "unrestricted MIDAS" (U-MIDAS); can be estimated by OLS. E.g.

$$y_t = \alpha + \beta y_{t-1} + \sum_{i=1}^{p} \gamma_i x_{\tau-i}$$

(where τ indicates "high-frequency time").

But more common to use something more parsimonious, for example:

- Normalized exponential Almon
- Normalized beta distribution (3 variants)
- (non-normalized) Almon polynomial

Parameterization math: an example

Normalized coefficient or weight i ($i = 1, \ldots, p$):

$$w_i = \frac{f(i, \theta)}{\sum_{i=1}^{p} f(i, \theta)}$$

such that the coefficients sum to unity.

In the exponential Almon case with k params the function $f(\cdot)$ is

$$f(i, \theta) = \exp\left(\sum_{j=1}^{k} \theta_j i^j\right)$$

In the usual two-parameter case we have

$$w_i = \frac{\exp\left(\theta_1 i + \theta_2 i^2\right)}{\sum_{i=1}^{p} \exp\left(\theta_1 i + \theta_2 i^2\right)}$$

with equal weighting when $\theta_1 = \theta_2 = 0$.

Parameterization functions

We offer: `mweights`, `mgradient`, `mlincomb`.

Examples:

```
matrix w = mweights(p, theta, 1)
```

Args: number of HF lags, hyperparameters, distribution code. Returns p-vector.

```
matrix g = mgradient(p, theta, 1)
```

Args: as mweights. Returns $p \times k$ matrix.

```
series mx = mlincomb(L, theta, 1)
```

is equivalent to

```
series mx = lincomb(L, mweights(nelem(L), theta, 1))
```

Parameterization functions

We offer: `mweights`, `mgradient`, `mlincomb`.

Examples:

```
matrix w = mweights(p, theta, 1)
```

Args: number of HF lags, hyperparameters, distribution code. Returns p-vector.

```
matrix g = mgradient(p, theta, 1)
```

Args: as mweights. Returns $p \times k$ matrix.

```
series mx = mlincomb(L, theta, 1)
```

is equivalent to

```
series mx = lincomb(L, mweights(nelem(L), theta, 1))
```

Parameterization functions

We offer: `mweights`, `mgradient`, `mlincomb`.

Examples:

```
matrix w = mweights(p, theta, 1)
```

Args: number of HF lags, hyperparameters, distribution code. Returns p-vector.

```
matrix g = mgradient(p, theta, 1)
```

Args: as mweights. Returns $p \times k$ matrix.

```
series mx = mlincomb(L, theta, 1)
```

is equivalent to

```
series mx = lincomb(L, mweights(nelem(L), theta, 1))
```

Parameterization functions

We offer: `mweights`, `mgradient`, `mlincomb`.

Examples:

```
matrix w = mweights(p, theta, 1)
```

Args: number of HF lags, hyperparameters, distribution code. Returns p-vector.

```
matrix g = mgradient(p, theta, 1)
```

Args: as mweights. Returns $p \times k$ matrix.

```
series mx = mlincomb(L, theta, 1)
```

is equivalent to

```
series mx = lincomb(L, mweights(nelem(L), theta, 1))
```

Uses of parameterization functions

Not required for estimation purposes if our built-in `midasreg` command (coming up!) meets your needs.

But useful if you want to roll your own MIDAS estimator.

Also useful if you want to explore the shapes of these functions. Example, for normalized beta:

```
matrix theta = {1,1}
matrix shapes = {}
loop for i=1..12
  theta[2] = i
  shapes ~= mweights(10, theta, 2)
endloop
shapes ~= seq(1,10)'
colnames(shapes, "1 2 3 4 5 6 7 8 9 10 11 12 lag")
gnuplot --matrix=shapes --with-lines --output=display \
{ set ylabel 'weight'; }
```

Uses of parameterization functions

Not required for estimation purposes if our built-in `midasreg` command (coming up!) meets your needs.

But useful if you want to roll your own MIDAS estimator.

Also useful if you want to explore the shapes of these functions. Example, for normalized beta:

```
matrix theta = {1,1}
matrix shapes = {}
loop for i=1..12
  theta[2] = i
  shapes ~= mweights(10, theta, 2)
endloop
shapes ~= seq(1,10)'
colnames(shapes, "1 2 3 4 5 6 7 8 9 10 11 12 lag")
gnuplot --matrix=shapes --with-lines --output=display \
{ set ylabel 'weight'; }
```

Uses of parameterization functions

Not required for estimation purposes if our built-in `midasreg` command (coming up!) meets your needs.

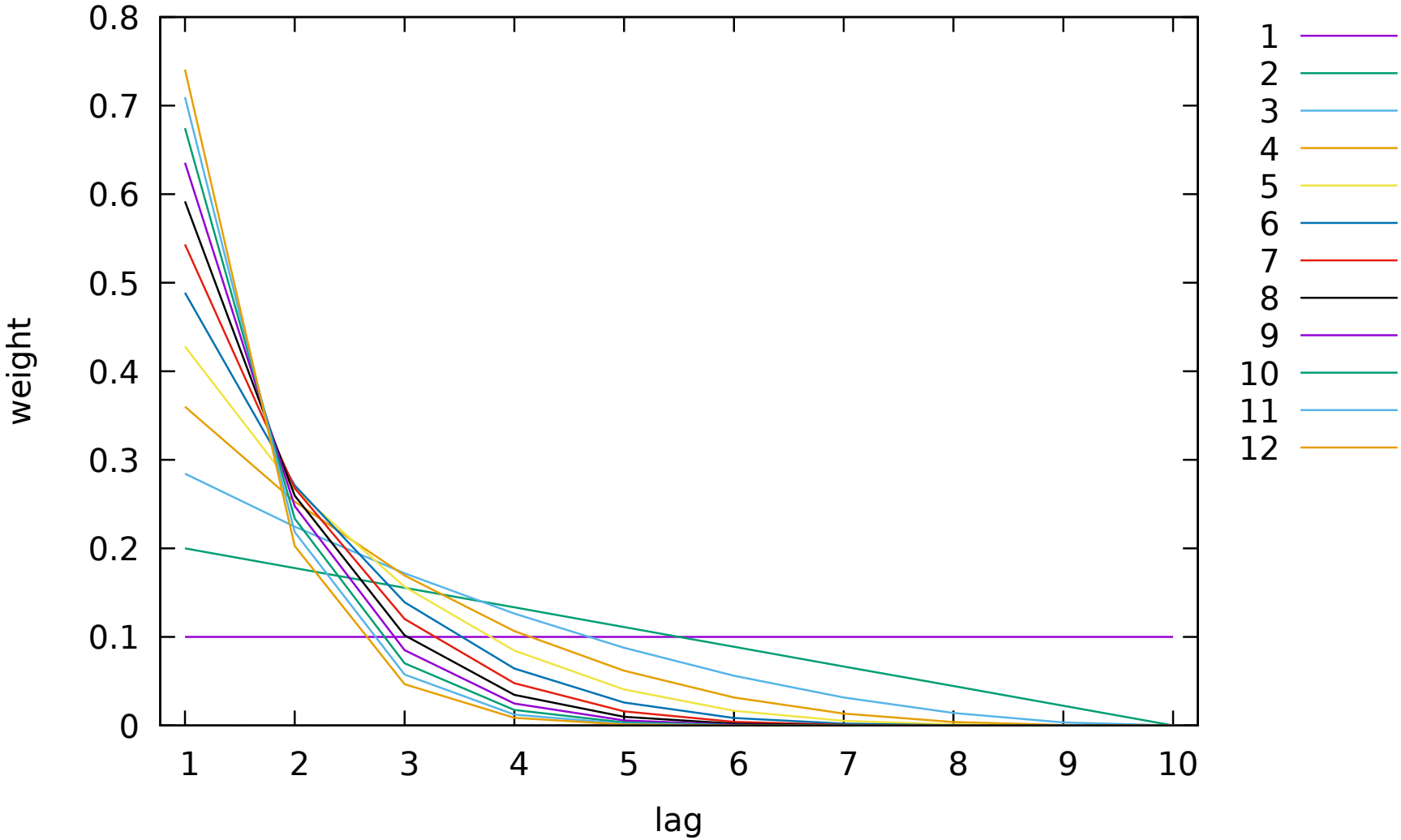
But useful if you want to roll your own MIDAS estimator.

Also useful if you want to explore the shapes of these functions. Example, for normalized beta:

```
matrix theta = {1,1}
matrix shapes = {}
loop for i=1..12
  theta[2] = i
  shapes ~= mweights(10, theta, 2)
endloop
shapes ~= seq(1,10)'
colnames(shapes, "1 2 3 4 5 6 7 8 9 10 11 12 lag")
gnuplot --matrix=shapes --with-lines --output=display \
{ set ylabel 'weight'; }
```

Beta shapes

$\theta_1 = 1$, θ_2 value shown in key; lag order 10.



Estimation via `midasreg`

The syntax of `midasreg`:

`midasreg` *depvar xlist* ; *midas-terms* [*options*]

midas-terms specifications:

1. mds(*mlist*, *minlag*, *maxlag*, *type*, *theta*)
2. mds(*mlist*, *minlag*, *maxlag*, 0)
3. mdsl(*llist*, *type*, *theta*)
4. mdsl(*llist*, 0)

Cases 1, 2: *mlist* is a MIDAS list (no lags included). Lags are generated automatically, governed by the *minlag* and *maxlag* arguments.

Cases 3, 4: *llist* already contains the required set of high-frequency lags.

Cases 2 and 4 are for U-MIDAS: initial θ is not needed.

Estimation via `midasreg`

The syntax of `midasreg`:

`midasreg` *depvar xlist* ; *midas-terms* [*options*]

midas-terms specifications:

1. `mds(`*mlist*`, `*minlag*`, `*maxlag*`, `*type*`, `*theta*`)`
2. `mds(`*mlist*`, `*minlag*`, `*maxlag*`, 0)`
3. `mdsl(`*llist*`, `*type*`, `*theta*`)`
4. `mdsl(`*llist*`, 0)`

Cases 1, 2: *mlist* is a MIDAS list (no lags included). Lags are generated automatically, governed by the *minlag* and *maxlag* arguments.

Cases 3, 4: *llist* already contains the required set of high-frequency lags.

Cases 2 and 4 are for U-MIDAS: initial θ is not needed.

Estimation via `midasreg`

The syntax of `midasreg`:

`midasreg` *depvar xlist* ; *midas-terms* [*options*]

midas-terms specifications:

1. `mds(`*mlist*`,` *minlag*`,` *maxlag*`,` *type*`,` *theta*`)`
2. `mds(`*mlist*`,` *minlag*`,` *maxlag*`, 0)`
3. `mdsl(`*llist*`,` *type*`,` *theta*`)`
4. `mdsl(`*llist*`, 0)`

Cases 1, 2: *mlist* is a MIDAS list (no lags included). Lags are generated automatically, governed by the *minlag* and *maxlag* arguments.

Cases 3, 4: *llist* already contains the required set of high-frequency lags.

Cases 2 and 4 are for U-MIDAS: initial θ is not needed.

Estimation via `midasreg`

The syntax of `midasreg`:

`midasreg` *depvar xlist* ; *midas-terms* [*options*]

midas-terms specifications:

1. mds(*mlist*, *minlag*, *maxlag*, *type*, *theta*)
2. mds(*mlist*, *minlag*, *maxlag*, 0)
3. mdsl(*llist*, *type*, *theta*)
4. mdsl(*llist*, 0)

Cases 1, 2: *mlist* is a MIDAS list (no lags included). Lags are generated automatically, governed by the *minlag* and *maxlag* arguments.

Cases 3, 4: *llist* already contains the required set of high-frequency lags.

Cases 2 and 4 are for U-MIDAS: initial θ is not needed.

Estimation via `midasreg`

The syntax of `midasreg`:

`midasreg` *depvar xlist* ; *midas-terms* [*options*]

midas-terms specifications:

1. `mds(`*mlist*`, `*minlag*`, `*maxlag*`, `*type*`, `*theta*`)`
2. `mds(`*mlist*`, `*minlag*`, `*maxlag*`, 0)`
3. `mdsl(`*llist*`, `*type*`, `*theta*`)`
4. `mdsl(`*llist*`, 0)`

Cases 1, 2: *mlist* is a MIDAS list (no lags included). Lags are generated automatically, governed by the *minlag* and *maxlag* arguments.

Cases 3, 4: *llist* already contains the required set of high-frequency lags.

Cases 2 and 4 are for U-MIDAS: initial θ is not needed.

Example of midasreg usage

Using the MIDAS dataset supplied with gretl, replicate one of Ghysels' Matlab examples.

```
open gdp_midas.gdt --quiet

# form the dependent variable
series dy = 100 * ldiff(qgdp)
# form list of high-frequency lagged log differences
list X = payems*
list dXL = hflags(3, 11, hfldiff(X, 100))

# estimation sample
smpl 1985:1 2009:1

print "normalized beta with zero last lag"
midasreg dy 0 dy(-1) ; mdsl(dXL, 2, {1,5})
```

Example of `midasreg` output

```
Model 1: MIDAS (NLS), using observations 1985:1-2009:1 (T = 97)
Using L-BFGS-B with conditional OLS
Dependent variable: dy

             estimate   std. error   t-ratio   p-value
  ---------------------------------------------------------
  const      0.665560    0.139647    4.766     7.00e-06 ***
  dy_1       0.284700    0.118466    2.403     0.0183   **

       MIDAS list dXL, high-frequency lags 3 to 11

  HF_slope   1.91207     0.574921    3.326     0.0013   ***
  Beta1      0.990377    0.106112    9.333     5.77e-15 ***
  Beta2      6.61573    17.1396      0.3860    0.7004

Mean dependent var   1.274925   S.D. dependent var   0.682517
Sum squared resid    29.64215   S.E. of regression   0.567624
R-squared            0.337155   Adjusted R-squared   0.308336
Log-likelihood      -80.13963   Akaike criterion     170.2793
Schwarz criterion    183.1528   Hannan-Quinn         175.4847
rho                 -0.036012   Durbin's h          -0.354681

GNR: R-squared = 5.77316e-15, max |t| = 5.6468e-07
Convergence seems to be reasonably complete
```

We can reproduce the MIDAS Matlab Toolbox results to at least 4 significant figures. . . apart from standard errors on the hyperparameters.

Example of `midasreg` output

```
Model 1: MIDAS (NLS), using observations 1985:1-2009:1 (T = 97)
Using L-BFGS-B with conditional OLS
Dependent variable: dy

             estimate   std. error   t-ratio   p-value
  ---------------------------------------------------------
  const      0.665560    0.139647    4.766     7.00e-06 ***
  dy_1       0.284700    0.118466    2.403     0.0183   **

       MIDAS list dXL, high-frequency lags 3 to 11

  HF_slope   1.91207     0.574921    3.326     0.0013   ***
  Beta1      0.990377    0.106112    9.333     5.77e-15 ***
  Beta2      6.61573    17.1396      0.3860    0.7004

Mean dependent var   1.274925   S.D. dependent var   0.682517
Sum squared resid    29.64215   S.E. of regression   0.567624
R-squared            0.337155   Adjusted R-squared   0.308336
Log-likelihood      -80.13963   Akaike criterion     170.2793
Schwarz criterion    183.1528   Hannan-Quinn         175.4847
rho                 -0.036012   Durbin's h          -0.354681

GNR: R-squared = 5.77316e-15, max |t| = 5.6468e-07
Convergence seems to be reasonably complete
```

We can reproduce the MIDAS Matlab Toolbox results to at least 4 significant figures. . . apart from standard errors on the hyperparameters.

Back-ends to `midasreg`

The `midasreg` command calls one of several possible estimation methods in the background, depending on the MIDAS specification(s).

If the only specification type is U-MIDAS, the method is OLS. Otherwise it is one of three variants of Nonlinear Least Squares.

- Levenberg–Marquardt. This is the back-end for gretl's `nls` command.
- L-BFGS-B with conditional OLS. L-BFGS is a "limited memory" version of the BFGS optimizer and the trailing "-B" means that it supports bounds on the parameters.
- Golden Section search with conditional OLS. This is a line search method, used only when there is a just a single hyperparameter to estimate.

Back-ends to `midasreg`

The `midasreg` command calls one of several possible estimation methods in the background, depending on the MIDAS specification(s).

If the only specification type is U-MIDAS, the method is OLS. Otherwise it is one of three variants of Nonlinear Least Squares.

- Levenberg–Marquardt. This is the back-end for gretl's `nls` command.
- L-BFGS-B with conditional OLS. L-BFGS is a "limited memory" version of the BFGS optimizer and the trailing "-B" means that it supports bounds on the parameters.
- Golden Section search with conditional OLS. This is a line search method, used only when there is a just a single hyperparameter to estimate.

Back-ends to `midasreg`

The `midasreg` command calls one of several possible estimation methods in the background, depending on the MIDAS specification(s).

If the only specification type is U-MIDAS, the method is OLS. Otherwise it is one of three variants of Nonlinear Least Squares.

- Levenberg–Marquardt. This is the back-end for gretl's `nls` command.
- L-BFGS-B with conditional OLS. L-BFGS is a "limited memory" version of the BFGS optimizer and the trailing "-B" means that it supports bounds on the parameters.
- Golden Section search with conditional OLS. This is a line search method, used only when there is a just a single hyperparameter to estimate.

Back-ends to `midasreg`

The `midasreg` command calls one of several possible estimation methods in the background, depending on the MIDAS specification(s).

If the only specification type is U-MIDAS, the method is OLS. Otherwise it is one of three variants of Nonlinear Least Squares.

- Levenberg–Marquardt. This is the back-end for gretl's `nls` command.
- L-BFGS-B with conditional OLS. L-BFGS is a "limited memory" version of the BFGS optimizer and the trailing "-B" means that it supports bounds on the parameters.
- Golden Section search with conditional OLS. This is a line search method, used only when there is a just a single hyperparameter to estimate.

Back-ends to `midasreg`

The `midasreg` command calls one of several possible estimation methods in the background, depending on the MIDAS specification(s).

If the only specification type is U-MIDAS, the method is OLS. Otherwise it is one of three variants of Nonlinear Least Squares.

- Levenberg–Marquardt. This is the back-end for gretl's `nls` command.
- L-BFGS-B with conditional OLS. L-BFGS is a "limited memory" version of the BFGS optimizer and the trailing "-B" means that it supports bounds on the parameters.
- Golden Section search with conditional OLS. This is a line search method, used only when there is a just a single hyperparameter to estimate.

Back-ends, continued

Levenberg–Marquardt is the default NLS method, but if the MIDAS specs include any beta variant or normalized exponential Almon we switch to L-BFGS-B, *unless* the user gives the `--levenberg` option.

Setting bounds on the hyperparameters via L-BFGS-B is handy: (a) the beta parameters must be non-negative; (b) we run into numerical problems if their values become too extreme.

"Conditional OLS" in the context of L-BFGS-B and line search: the search algorithm is responsible for optimizing the MIDAS hyperparameter(s) only. When the algorithm calls for calculation of SSR given θ we optimize all remaining parameters via OLS.

Back-ends, continued

Levenberg–Marquardt is the default NLS method, but if the MIDAS specs include any beta variant or normalized exponential Almon we switch to L-BFGS-B, *unless* the user gives the `--levenberg` option.

Setting bounds on the hyperparameters via L-BFGS-B is handy: (a) the beta parameters must be non-negative; (b) we run into numerical problems if their values become too extreme.

"Conditional OLS" in the context of L-BFGS-B and line search: the search algorithm is responsible for optimizing the MIDAS hyperparameter(s) only. When the algorithm calls for calculation of SSR given θ we optimize all remaining parameters via OLS.

Back-ends, continued

Levenberg–Marquardt is the default NLS method, but if the MIDAS specs include any beta variant or normalized exponential Almon we switch to L-BFGS-B, *unless* the user gives the `--levenberg` option.

Setting bounds on the hyperparameters via L-BFGS-B is handy: (a) the beta parameters must be non-negative; (b) we run into numerical problems if their values become too extreme.

"Conditional OLS" in the context of L-BFGS-B and line search: the search algorithm is responsible for optimizing the MIDAS hyperparameter(s) only. When the algorithm calls for calculation of SSR given θ we optimize all remaining parameters via OLS.

Transition. . .

About to end the exposition of gretl's MIDAS functionality.

But take a peek at the midasreg GUI first.

Now we'll move on to assessing MIDAS as a forecasting methodology.

Any questions first?

Transition. . .

About to end the exposition of gretl's MIDAS functionality.

But take a peek at the midasreg GUI first.

Now we'll move on to assessing MIDAS as a forecasting methodology.

Any questions first?

Questions

- Given the timing with which relevant data become available, what are the options for forecasting at different horizons, and what are the implications for the lag structure of the models one may use?
- What choices of high-frequency data and MIDAS parameterization give the best forecasting performance?
- How do MIDAS-based forecasts compare with simpler methods that use data of a single frequency?

We'll focus on forecasting (the log difference of) US real GDP.

Questions

- Given the timing with which relevant data become available, what are the options for forecasting at different horizons, and what are the implications for the lag structure of the models one may use?
- What choices of high-frequency data and MIDAS parameterization give the best forecasting performance?
- How do MIDAS-based forecasts compare with simpler methods that use data of a single frequency?

We'll focus on forecasting (the log difference of) US real GDP.

Questions

- Given the timing with which relevant data become available, what are the options for forecasting at different horizons, and what are the implications for the lag structure of the models one may use?
- What choices of high-frequency data and MIDAS parameterization give the best forecasting performance?
- How do MIDAS-based forecasts compare with simpler methods that use data of a single frequency?

We'll focus on forecasting (the log difference of) US real GDP.

Questions

- Given the timing with which relevant data become available, what are the options for forecasting at different horizons, and what are the implications for the lag structure of the models one may use?
- What choices of high-frequency data and MIDAS parameterization give the best forecasting performance?
- How do MIDAS-based forecasts compare with simpler methods that use data of a single frequency?

We'll focus on forecasting (the log difference of) US real GDP.

Data timing

Define the “data lag” for a given series as the lag between the end of a period and the first publication of data pertaining to that period.

Approximate data lags for some commonly referenced US macro time series:

series	source	frequency	approx lag
CPI	BLS	monthly	2 weeks
PAYEMS	BLS	monthly	1 week
INDPRO	Fed	monthly	2 weeks
GDP	BEA	quarterly	4 weeks

Data timing

Define the “data lag” for a given series as the lag between the end of a period and the first publication of data pertaining to that period.

Approximate data lags for some commonly referenced US macro time series:

series	source	frequency	approx lag
CPI	BLS	monthly	2 weeks
PAYEMS	BLS	monthly	1 week
INDPRO	Fed	monthly	2 weeks
GDP	BEA	quarterly	4 weeks

Data arrival schedule for a given quarter

Take quarter Q_t as "the present" (lag 0); "h" against a lag indicates a high-frequency lag.

Month	end week	latest data	lag
1	1	PAYEMS Q_{t-1} month 3	1h
	2	INDPRO Q_{t-1} month 3	1h
	4	est. 1, GDP Q_{t-1}	1
2	1	PAYEMS Q_t, month 1	0h
	2	INDPRO Q_t, month 1	0h
	4	est. 2, GDP Q_{t-1}	1
3	1	PAYEMS Q_t, month 2	−1h
	2	INDPRO Q_t, month 2	−1h
	4	est. 3, GDP Q_{t-1}	1

Lag structure

Consider an ADL-MIDAS model (in log differences) for GDP, using the first lag of GDP and HF lags 1 to p of INDPRO:

$$y_t = \alpha + \beta y_{t-1} + \gamma W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-p}; \theta) + \varepsilon_t$$

The forecast from this model is then

$$\hat{y}_t = \hat{\alpha} + \hat{\beta} y_{t-1} + \hat{\gamma} W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-p}; \hat{\theta})$$

Assume that to generate a forecast we require actual published values for all the regressors.

Then we won't be able to generate a forecast of quarter Q_t's GDP until the end of week 4 of the quarter, when the advance estimate of the prior quarter's GDP becomes available.

Lag structure

Consider an ADL-MIDAS model (in log differences) for GDP, using the first lag of GDP and HF lags 1 to p of INDPRO:

$$y_t = \alpha + \beta y_{t-1} + \gamma W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-p}; \theta) + \varepsilon_t$$

The forecast from this model is then

$$\hat{y}_t = \hat{\alpha} + \hat{\beta} y_{t-1} + \hat{\gamma} W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-p}; \hat{\theta})$$

Assume that to generate a forecast we require actual published values for all the regressors.

Then we won't be able to generate a forecast of quarter Q_t's GDP until the end of week 4 of the quarter, when the advance estimate of the prior quarter's GDP becomes available.

Lag structure

Consider an ADL-MIDAS model (in log differences) for GDP, using the first lag of GDP and HF lags 1 to p of INDPRO:

$$y_t = \alpha + \beta y_{t-1} + \gamma W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-p}; \theta) + \varepsilon_t$$

The forecast from this model is then

$$\hat{y}_t = \hat{\alpha} + \hat{\beta} y_{t-1} + \hat{\gamma} W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-p}; \hat{\theta})$$

Assume that to generate a forecast we require actual published values for all the regressors.

Then we won't be able to generate a forecast of quarter Q_t's GDP until the end of week 4 of the quarter, when the advance estimate of the prior quarter's GDP becomes available.

Lag structure

Consider an ADL-MIDAS model (in log differences) for GDP, using the first lag of GDP and HF lags 1 to p of INDPRO:

$$y_t = \alpha + \beta y_{t-1} + \gamma W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-p}; \theta) + \varepsilon_t$$

The forecast from this model is then

$$\hat{y}_t = \hat{\alpha} + \hat{\beta} y_{t-1} + \hat{\gamma} W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-p}; \hat{\theta})$$

Assume that to generate a forecast we require actual published values for all the regressors.

Then we won't be able to generate a forecast of quarter Q_t's GDP until the end of week 4 of the quarter, when the advance estimate of the prior quarter's GDP becomes available.

Updated nowcast

An updated nowcast could be produced at various points during the quarter.

A fitted value could be recalculated using the revised GDP figures for Q_{t-1} available towards the end of months 2 and 3.

A second model whose HF lags start at 0 could produce a nowcast incorporating the new INDPRO information that arrives in month 2:

$$y_t = \alpha_1 + \beta_1 y_{t-1} + \gamma_1 W(x_\tau, x_{\tau-1}, \ldots, x_{\tau-p+1}; \theta_1) + \eta_t$$

And a model whose HF lags start at -1 could use INDPRO from month 3 to produce a further update.

Updated nowcast

An updated nowcast could be produced at various points during the quarter.

A fitted value could be recalculated using the revised GDP figures for Q_{t-1} available towards the end of months 2 and 3.

A second model whose HF lags start at 0 could produce a nowcast incorporating the new INDPRO information that arrives in month 2:

$$y_t = \alpha_1 + \beta_1 y_{t-1} + \gamma_1 W(x_\tau, x_{\tau-1}, \ldots, x_{\tau-p+1}; \theta_1) + \eta_t$$

And a model whose HF lags start at -1 could use INDPRO from month 3 to produce a further update.

Updated nowcast

An updated nowcast could be produced at various points during the quarter.

A fitted value could be recalculated using the revised GDP figures for Q_{t-1} available towards the end of months 2 and 3.

A second model whose HF lags start at 0 could produce a nowcast incorporating the new INDPRO information that arrives in month 2:

$$y_t = \alpha_1 + \beta_1 y_{t-1} + \gamma_1 W(x_\tau, x_{\tau-1}, \ldots, x_{\tau-p+1}; \theta_1) + \eta_t$$

And a model whose HF lags start at -1 could use INDPRO from month 3 to produce a further update.

Updated nowcast

An updated nowcast could be produced at various points during the quarter.

A fitted value could be recalculated using the revised GDP figures for Q_{t-1} available towards the end of months 2 and 3.

A second model whose HF lags start at 0 could produce a nowcast incorporating the new INDPRO information that arrives in month 2:

$$y_t = \alpha_1 + \beta_1 y_{t-1} + \gamma_1 W(x_\tau, x_{\tau-1}, \ldots, x_{\tau-p+1}; \theta_1) + \eta_t$$

And a model whose HF lags start at -1 could use INDPRO from month 3 to produce a further update.

Chained forecasts?

We're forecasting GDP using lagged GDP, so in principle the forecast could be chained: based on observed GDP for Q_{t-1} we could produce an estimate for Q_t then use this in the equation for Q_{t+1}, before the Q_t datum is published.

Expanding the forecast horizon in this way requires that

- we somehow obtain forecasts for the high frequency data too, or
- we revise the model to employ less recent lags of the high frequency data.

Take up the second possibility: in week 4 of Q_t we can form a first nowcast of Q_t's GDP, but at this point the most recent INDPRO datum is from month 3 of Q_{t-1}, which is HF lag 4 relative to the equation for Q_{t+1}.

Chained forecasts?

We're forecasting GDP using lagged GDP, so in principle the forecast could be chained: based on observed GDP for Q_{t-1} we could produce an estimate for Q_t then use this in the equation for Q_{t+1}, before the Q_t datum is published.

Expanding the forecast horizon in this way requires that

- we somehow obtain forecasts for the high frequency data too, or
- we revise the model to employ less recent lags of the high frequency data.

Take up the second possibility: in week 4 of Q_t we can form a first nowcast of Q_t's GDP, but at this point the most recent INDPRO datum is from month 3 of Q_{t-1}, which is HF lag 4 relative to the equation for Q_{t+1}.

Chained forecasts?

We're forecasting GDP using lagged GDP, so in principle the forecast could be chained: based on observed GDP for Q_{t-1} we could produce an estimate for Q_t then use this in the equation for Q_{t+1}, before the Q_t datum is published.

Expanding the forecast horizon in this way requires that

- we somehow obtain forecasts for the high frequency data too, or
- we revise the model to employ less recent lags of the high frequency data.

Take up the second possibility: in week 4 of Q_t we can form a first nowcast of Q_t's GDP, but at this point the most recent INDPRO datum is from month 3 of Q_{t-1}, which is HF lag 4 relative to the equation for Q_{t+1}.

Chaining, continued

The model for forecasting GDP "truly ahead" (as opposed to a nowcast) would have to be estimated with a minimum HF lag of 4:

$$y_t = \alpha_2 + \beta_2 y_{t-1} + \gamma_2 W(x_{\tau-4}, x_{\tau-5}, \ldots, x_{\tau-q}; \theta_2) + \nu_t$$

We could then calculate at the end of month 1 of Q_t

$$\hat{y}_{t+1} = \hat{\alpha}_2 + \hat{\beta}_2 \tilde{y}_t + \hat{\gamma}_2 W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-q+3}; \hat{\theta}_2)$$

with $\tilde{y}_t$ obtained as a fitted value from the basic forecasting equation.

Not sure if this is actually of practical interest. ;-)

Chaining, continued

The model for forecasting GDP "truly ahead" (as opposed to a nowcast) would have to be estimated with a minimum HF lag of 4:

$$y_t = \alpha_2 + \beta_2 y_{t-1} + \gamma_2 W(x_{\tau-4}, x_{\tau-5}, \ldots, x_{\tau-q}; \theta_2) + \nu_t$$

We could then calculate at the end of month 1 of Q_t

$$\hat{y}_{t+1} = \hat{\alpha}_2 + \hat{\beta}_2 \tilde{y}_t + \hat{\gamma}_2 W(x_{\tau-1}, x_{\tau-2}, \ldots, x_{\tau-q+3}; \hat{\theta}_2)$$

with $\tilde{y}_t$ obtained as a fitted value from the basic forecasting equation.

Not sure if this is actually of practical interest. ;-)

MIDAS Matlab Toolbox examples

The MIDAS Matlab Toolbox ADL examples forecast the log difference of GDP using the first lag of the dependent variable and HF lags 3 to 11 of the log difference of monthly payroll employment.

These forecasts are static. So forecasts (nowcasts) cannot be produced until 4 weeks into the quarter. Then why use $x_{\tau-3}$ as the most recent monthly lag? (A published value for $x_{\tau-1}$ will surely be available.)

No call to be too pedantic. But we have good reason to consider alternative lag schemes in the experiments that follow.

MIDAS Matlab Toolbox examples

The MIDAS Matlab Toolbox ADL examples forecast the log difference of GDP using the first lag of the dependent variable and HF lags 3 to 11 of the log difference of monthly payroll employment.

These forecasts are static. So forecasts (nowcasts) cannot be produced until 4 weeks into the quarter. Then why use $x_{\tau-3}$ as the most recent monthly lag? (A published value for $x_{\tau-1}$ will surely be available.)

No call to be too pedantic. But we have good reason to consider alternative lag schemes in the experiments that follow.

MIDAS Matlab Toolbox examples

The MIDAS Matlab Toolbox ADL examples forecast the log difference of GDP using the first lag of the dependent variable and HF lags 3 to 11 of the log difference of monthly payroll employment.

These forecasts are static. So forecasts (nowcasts) cannot be produced until 4 weeks into the quarter. Then why use $x_{\tau-3}$ as the most recent monthly lag? (A published value for $x_{\tau-1}$ will surely be available.)

No call to be too pedantic. But we have good reason to consider alternative lag schemes in the experiments that follow.

Forecast comparisons

We compare forecasts along 5 dimensions. This invites combinatorial explosion. We limit ourselves to relatively few "tics" in most of the dimensions.

High-frequency regressor. We look at two candidate monthly series, the Fed's Index of Industrial Production (INDPRO) and non-farm Payroll Employment (PAYEMS).

Parameterization. We start with four MIDAS parameterizations and two single-frequency alternatives. The MIDAS variants are:

Beta 2	Two-parameter normalized beta
Beta 1	As Beta 2 but with θ_1 clamped at 1.0
NEAlmon	Normalized exponential Almon, 2 parameters
Almon poly	Almon polynomial of order 4

Forecast comparisons

We compare forecasts along 5 dimensions. This invites combinatorial explosion. We limit ourselves to relatively few "tics" in most of the dimensions.

High-frequency regressor. We look at two candidate monthly series, the Fed's Index of Industrial Production (INDPRO) and non-farm Payroll Employment (PAYEMS).

Parameterization. We start with four MIDAS parameterizations and two single-frequency alternatives. The MIDAS variants are:

Beta 2	Two-parameter normalized beta
Beta 1	As Beta 2 but with θ_1 clamped at 1.0
NEAlmon	Normalized exponential Almon, 2 parameters
Almon poly	Almon polynomial of order 4

Forecast comparisons

We compare forecasts along 5 dimensions. This invites combinatorial explosion. We limit ourselves to relatively few "tics" in most of the dimensions.

High-frequency regressor. We look at two candidate monthly series, the Fed's Index of Industrial Production (INDPRO) and non-farm Payroll Employment (PAYEMS).

Parameterization. We start with four MIDAS parameterizations and two single-frequency alternatives. The MIDAS variants are:

Beta 2	Two-parameter normalized beta
Beta 1	As Beta 2 but with θ_1 clamped at 1.0
NEAlmon	Normalized exponential Almon, 2 parameters
Almon poly	Almon polynomial of order 4

Comparison dimensions, continued

The single-frequency alternatives are:

AR(1)	OLS with regressors constant and y_{t-1}
ARMA(1,1)	Exact ML, including a constant

MIDAS lags. We fix on 10 lags of the high-frequency variable, but we compare results between lags 1 to 10 and lags 0 to 9.

Estimation sample size (T). In general, the more data the better. But if there are structural breaks or structural drift then a shorter sample may yield more accurate forecasts. T varies between 60 and 120 quarters.

Forecast start date. Results are likely to differ depending on the particular historical stretch of data. We initially use four different starting points for forecasts: 2000Q1, 2005Q1, 2010Q1 and 2015Q1.

Comparison dimensions, continued

The single-frequency alternatives are:

AR(1)	OLS with regressors constant and y_{t-1}
ARMA(1,1)	Exact ML, including a constant

MIDAS lags. We fix on 10 lags of the high-frequency variable, but we compare results between lags 1 to 10 and lags 0 to 9.

Estimation sample size (T). In general, the more data the better. But if there are structural breaks or structural drift then a shorter sample may yield more accurate forecasts. T varies between 60 and 120 quarters.

Forecast start date. Results are likely to differ depending on the particular historical stretch of data. We initially use four different starting points for forecasts: 2000Q1, 2005Q1, 2010Q1 and 2015Q1.

Comparison dimensions, continued

The single-frequency alternatives are:

AR(1)	OLS with regressors constant and y_{t-1}
ARMA(1,1)	Exact ML, including a constant

MIDAS lags. We fix on 10 lags of the high-frequency variable, but we compare results between lags 1 to 10 and lags 0 to 9.

Estimation sample size (T). In general, the more data the better. But if there are structural breaks or structural drift then a shorter sample may yield more accurate forecasts. T varies between 60 and 120 quarters.

Forecast start date. Results are likely to differ depending on the particular historical stretch of data. We initially use four different starting points for forecasts: 2000Q1, 2005Q1, 2010Q1 and 2015Q1.

Comparison dimensions, continued

The single-frequency alternatives are:

AR(1)	OLS with regressors constant and y_{t-1}
ARMA(1,1)	Exact ML, including a constant

MIDAS lags. We fix on 10 lags of the high-frequency variable, but we compare results between lags 1 to 10 and lags 0 to 9.

Estimation sample size (T). In general, the more data the better. But if there are structural breaks or structural drift then a shorter sample may yield more accurate forecasts. T varies between 60 and 120 quarters.

Forecast start date. Results are likely to differ depending on the particular historical stretch of data. We initially use four different starting points for forecasts: 2000Q1, 2005Q1, 2010Q1 and 2015Q1.

Data: realtime or hindsight?

Realtime: assume that the econometrician at time t had access only to data actually published at time $s \leq t$.

Hindsight: hypothetically endow the econometrician at time t with the current best estimate of quantities dated $s \leq t$.

Realtime is more complicated: have to assemble several archival datasets. (Though gretl can manage this: see `midas-supp.pdf` for details.)

Hindsight easier, and *perhaps* it's helpful to "net out" noise due to contemporaneous measurement error when comparing forecasting methods?

Data: realtime or hindsight?

Realtime: assume that the econometrician at time t had access only to data actually published at time $s \leq t$.

Hindsight: hypothetically endow the econometrician at time t with the current best estimate of quantities dated $s \leq t$.

Realtime is more complicated: have to assemble several archival datasets. (Though gretl can manage this: see `midas-supp.pdf` for details.)

Hindsight easier, and *perhaps* it's helpful to "net out" noise due to contemporaneous measurement error when comparing forecasting methods?

Data: realtime or hindsight?

Realtime: assume that the econometrician at time t had access only to data actually published at time $s \leq t$.

Hindsight: hypothetically endow the econometrician at time t with the current best estimate of quantities dated $s \leq t$.

Realtime is more complicated: have to assemble several archival datasets. (Though gretl can manage this: see `midas-supp.pdf` for details.)

Hindsight easier, and *perhaps* it's helpful to "net out" noise due to contemporaneous measurement error when comparing forecasting methods?

Data: realtime or hindsight?

Realtime: assume that the econometrician at time t had access only to data actually published at time $s \leq t$.

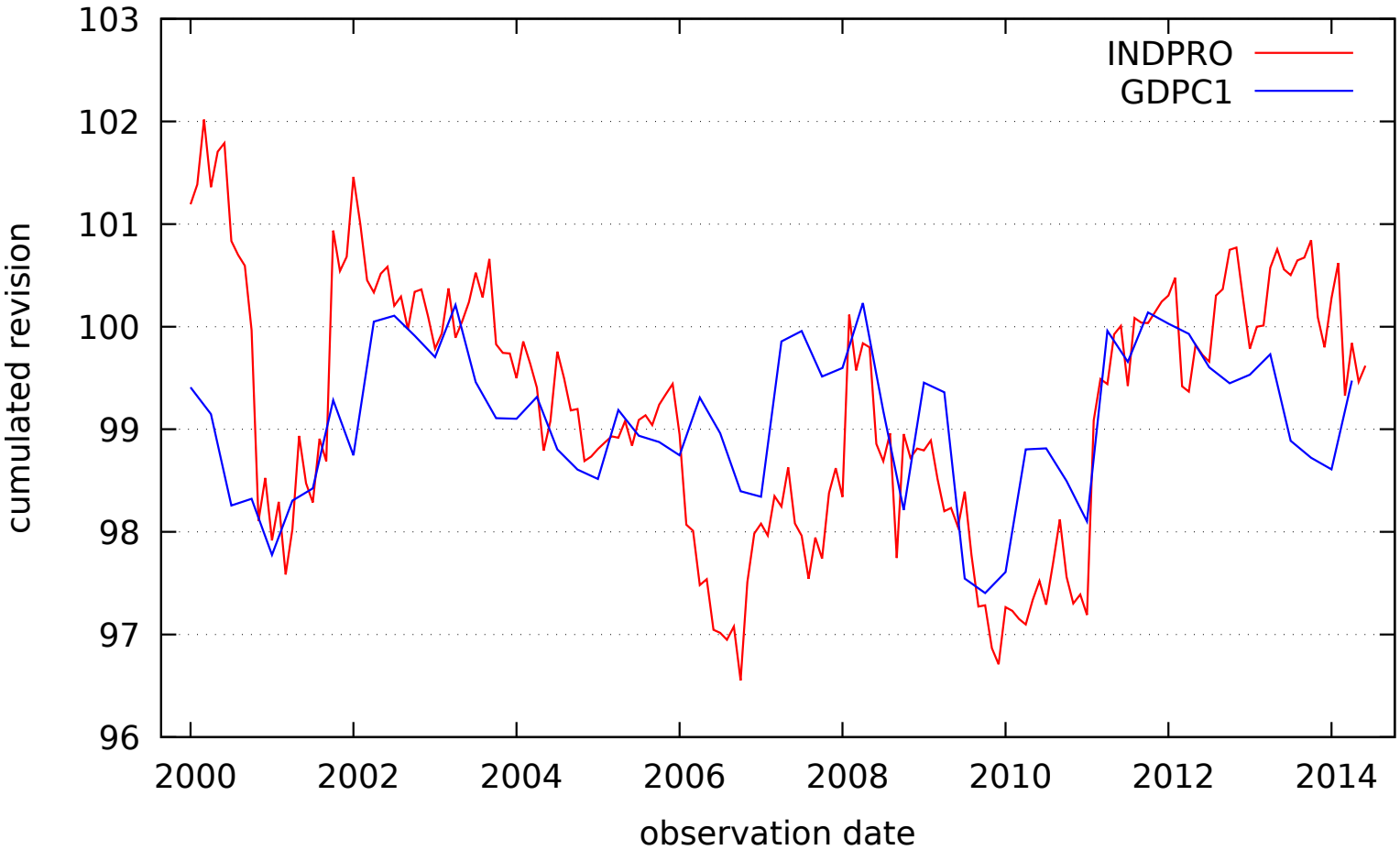
Hindsight: hypothetically endow the econometrician at time t with the current best estimate of quantities dated $s \leq t$.

Realtime is more complicated: have to assemble several archival datasets. (Though gretl can manage this: see `midas-supp.pdf` for details.)

Hindsight easier, and *perhaps* it's helpful to "net out" noise due to contemporaneous measurement error when comparing forecasting methods?

Possibly interesting?

$$\text{cumulated revision}_t = 100 \times y_{t,\tau=26} / y_{t,\tau=1}$$



Forecast assessment plots: procedure

- Choose a particular high-frequency predictor.
- Choose a particular MIDAS lag set.
- Choose a forecast start date.

Run an iteration across sample size: for each T, estimate each of the six models mentioned above, generate 8 static forecasts, and record the RMSE. Each plot shows RMSE against sample size.

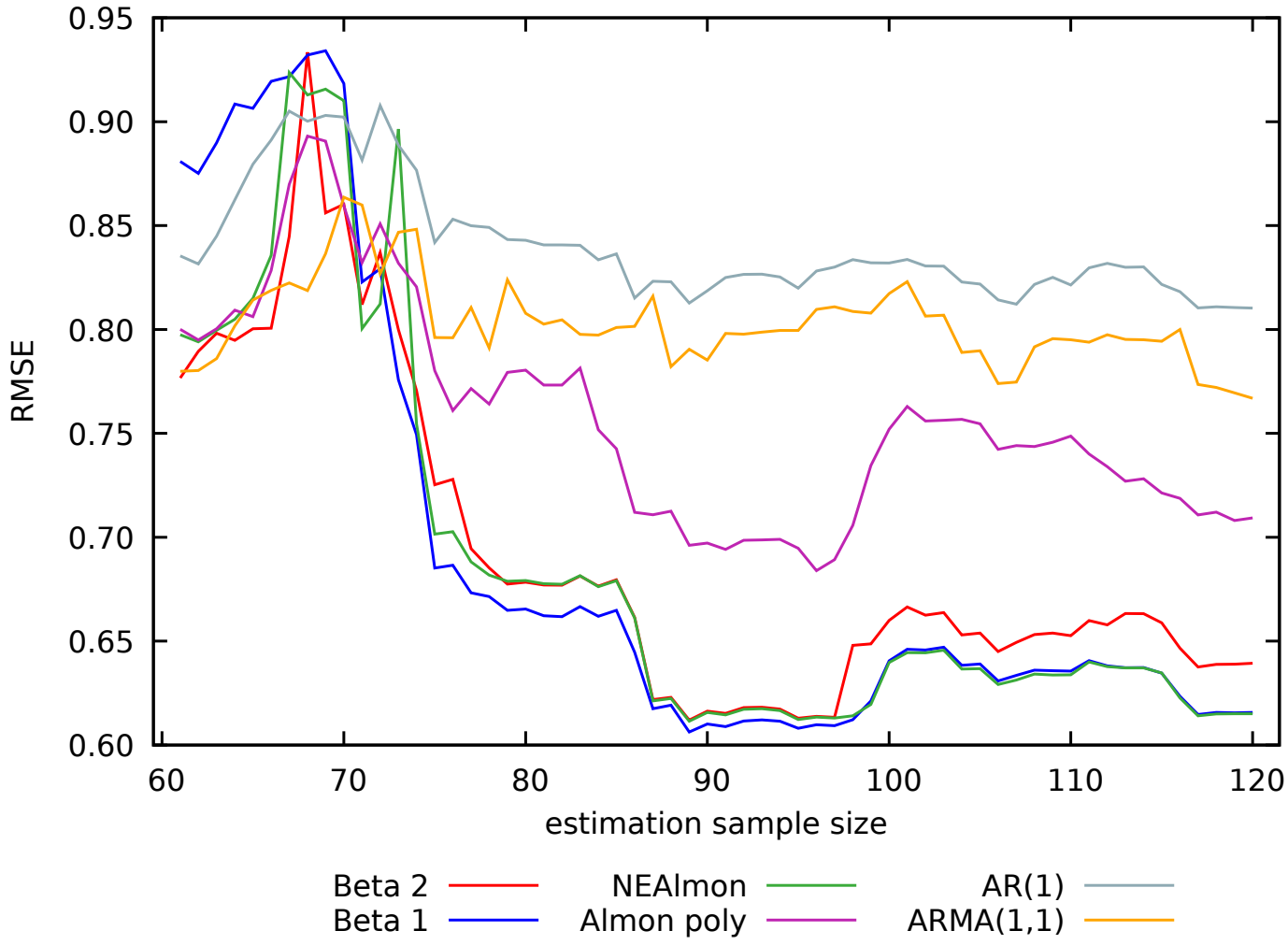
Forecast assessment plots: procedure

- Choose a particular high-frequency predictor.
- Choose a particular MIDAS lag set.
- Choose a forecast start date.

Run an iteration across sample size: for each T, estimate each of the six models mentioned above, generate 8 static forecasts, and record the RMSE. Each plot shows RMSE against sample size.

Forecast assessment plot: example

HF regressor INDPRO; HF lags 1 to 10; forecast start 2000Q1.



Murkier

Results become less clear when we explore the various dimensions.

[Audience participation!]

A different cut

We try reformulating the experiment such that it's easier to produce numerical figures of merit.

- Fix on $T = 90$ observations for estimation.
- Drop the riskier MIDAS variants, Beta 2 and Almon poly.
- Advance the forecast target quarter-by-quarter from 2000Q1 to 2016Q4, in each case re-estimating the models and generating a single forecast.
- Calculate Mean Absolute Error and Mean Square Error for the series of 68 forecasts, per model.

A different cut

We try reformulating the experiment such that it's easier to produce numerical figures of merit.

- Fix on $T = 90$ observations for estimation.
- Drop the riskier MIDAS variants, Beta 2 and Almon poly.
- Advance the forecast target quarter-by-quarter from 2000Q1 to 2016Q4, in each case re-estimating the models and generating a single forecast.
- Calculate Mean Absolute Error and Mean Square Error for the series of 68 forecasts, per model.

A different cut

We try reformulating the experiment such that it's easier to produce numerical figures of merit.

- Fix on $T = 90$ observations for estimation.
- Drop the riskier MIDAS variants, Beta 2 and Almon poly.
- Advance the forecast target quarter-by-quarter from 2000Q1 to 2016Q4, in each case re-estimating the models and generating a single forecast.
- Calculate Mean Absolute Error and Mean Square Error for the series of 68 forecasts, per model.

A different cut

We try reformulating the experiment such that it's easier to produce numerical figures of merit.

- Fix on $T = 90$ observations for estimation.
- Drop the riskier MIDAS variants, Beta 2 and Almon poly.
- Advance the forecast target quarter-by-quarter from 2000Q1 to 2016Q4, in each case re-estimating the models and generating a single forecast.
- Calculate Mean Absolute Error and Mean Square Error for the series of 68 forecasts, per model.

A different cut

We try reformulating the experiment such that it's easier to produce numerical figures of merit.

- Fix on $T = 90$ observations for estimation.
- Drop the riskier MIDAS variants, Beta 2 and Almon poly.
- Advance the forecast target quarter-by-quarter from 2000Q1 to 2016Q4, in each case re-estimating the models and generating a single forecast.
- Calculate Mean Absolute Error and Mean Square Error for the series of 68 forecasts, per model.

Results

HF series INDPRO

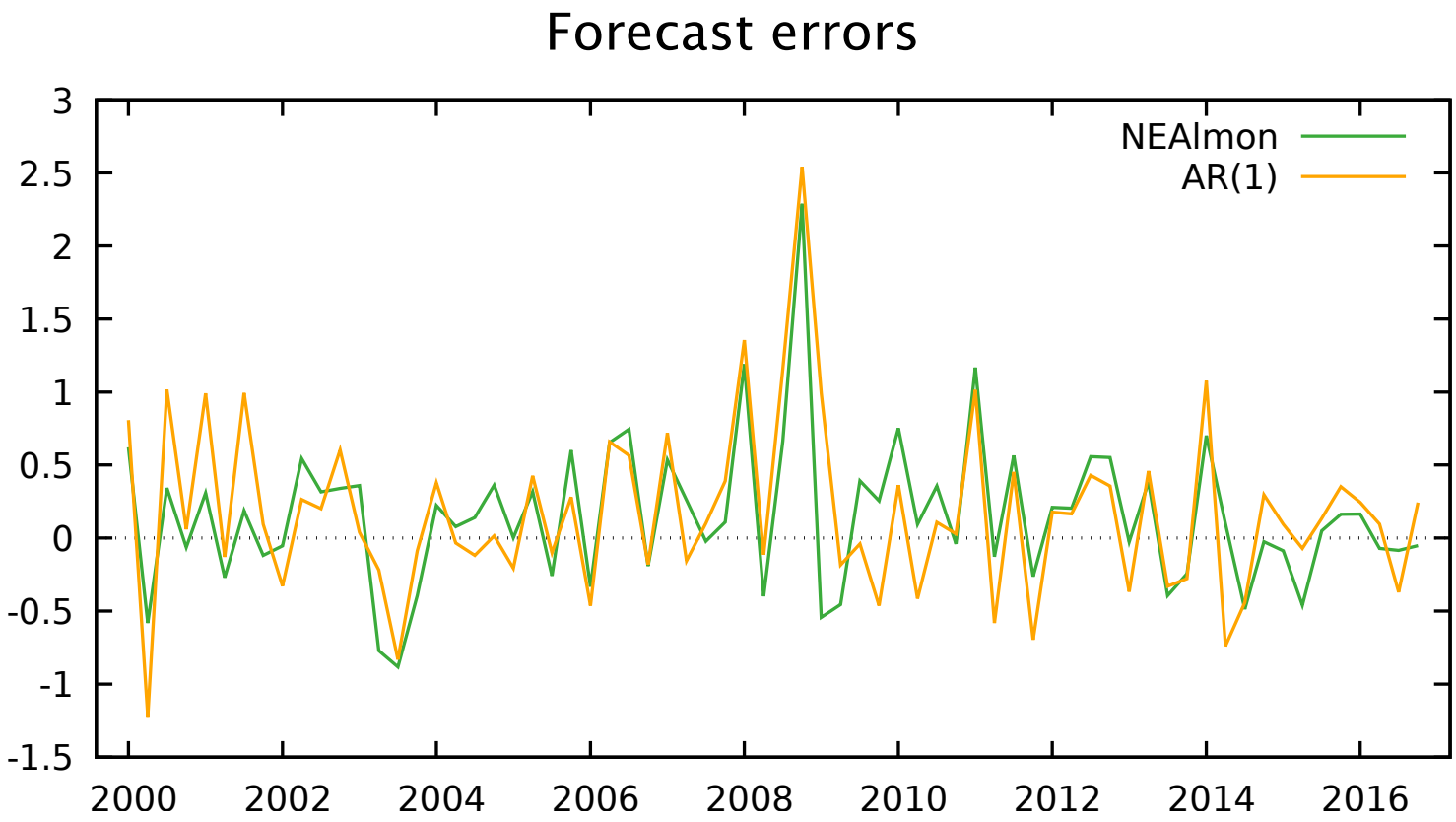
HF lags		Beta 1	NEAlmon	AR(1)	ARMA(1,1)	$t(67)$
0 to 9	MAE	0.3788	0.3758	0.4399	0.4445	−1.805
	MSE	0.2648	0.2648	0.3713	0.3590	−2.023
1 to 10	MAE	0.3935	0.3977	0.4399	0.4445	−1.581
	MSE	0.2874	0.2946	0.3713	0.3590	−1.953
3 to 11	MAE	0.4493	0.4594	0.4399	0.4445	0.697
	MSE	0.3867	0.3812	0.3713	0.3590	1.345

HF series PAYEMS

HF lags		Beta 1	NEAlmon	AR(1)	ARMA(1,1)	$t(67)$
0 to 9	MAE	0.4404	0.4400	0.4399	0.4445	0.004
	MSE	0.3124	0.3117	0.3713	0.3590	−0.131
1 to 10	MAE	0.4291	0.4303	0.4399	0.4445	−0.305
	MSE	0.3243	0.3286	0.3713	0.3590	−0.537
3 to 11	MAE	0.4502	0.4560	0.4399	0.4445	0.515
	MSE	0.3689	0.3803	0.3713	0.3590	0.439

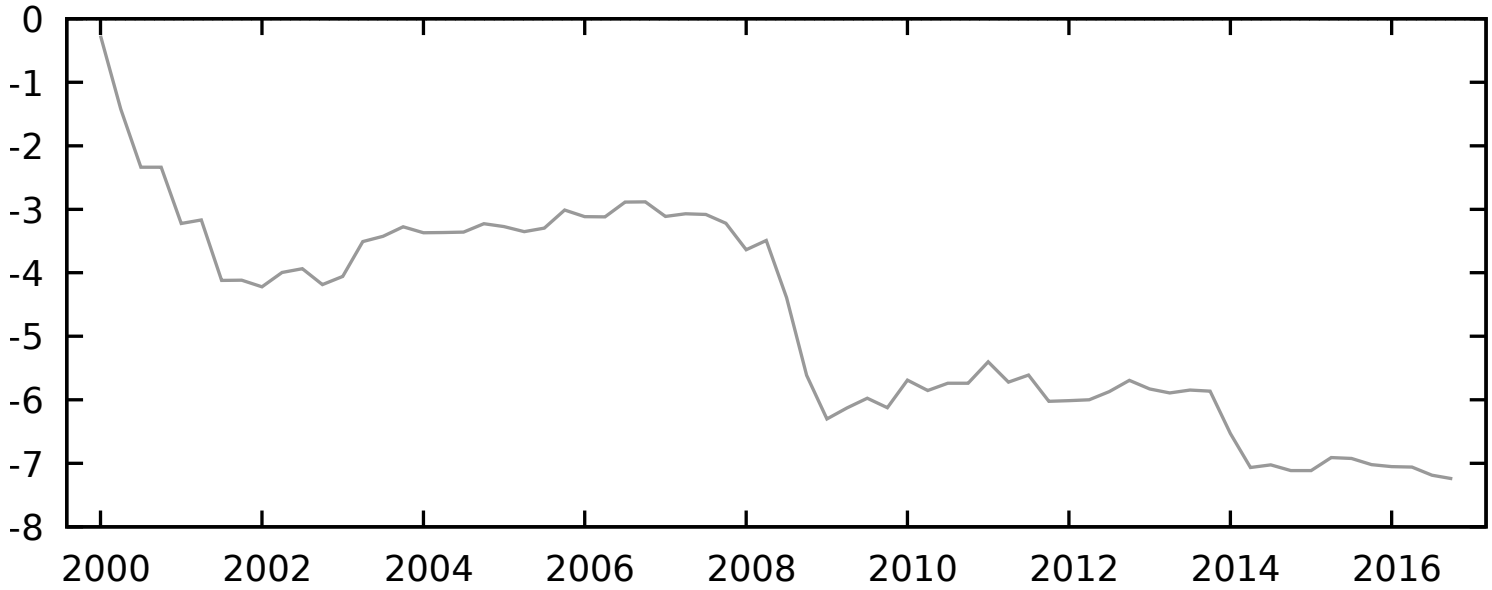
Another view

Normalized exponential Almon using HF lags 0 to 9 of INDPRO, versus AR(1).



Another view, contd.

Cumulation of difference in absolute errors, NEAlmon − AR(1).



Would this convince a skeptic?

It seems that the relative performance of monthly industrial production in forecasting US GDP in the 21st century should give some grounds for considering MIDAS as a live alternative.

But was this example cherry-picked?

Would be interesting to see if these findings are confirmed or thrown in doubt by examination of European macroeconomic data.

Would this convince a skeptic?

It seems that the relative performance of monthly industrial production in forecasting US GDP in the 21st century should give some grounds for considering MIDAS as a live alternative.

But was this example cherry-picked?

Would be interesting to see if these findings are confirmed or thrown in doubt by examination of European macroeconomic data.

Would this convince a skeptic?

It seems that the relative performance of monthly industrial production in forecasting US GDP in the 21st century should give some grounds for considering MIDAS as a live alternative.

But was this example cherry-picked?

Would be interesting to see if these findings are confirmed or thrown in doubt by examination of European macroeconomic data.

Thank you for your attention.