

APPENDICES

APPENDIX A

PARAMETER SETTINGS

A.1 ESTIMATED PARAMETERS OF THE MIDAS MODEL

Table A.1 Estimated Parameters of the MIDAS Model

Parameter	Estimated Value
β_0	128.349
β_1	2.954
β_2	9.910
$\nu_1 = (\nu_{11}, \nu_{12})$	(5.117, 35.427)
$\nu_2 = (\nu_{21}, \nu_{22})$	(8.947, 41.842)

Note: ν_{11} and ν_{12} denote the first and second parameters of the Exponential Almon lag polynomial for the DXY index variable, respectively. Similarly, ν_{21} and ν_{22} denote the first and second parameters of the Exponential Almon lag polynomial for the OIL variable, respectively.

Table A.2 MIDAS Lag Weights Based on the Estimated Exponential Almon Polynomial

k	DXY Weight $w_{1,k}(\nu_1)$	OIL Weight $w_{2,k}(\nu_2)$
1	0.000	0.000
2	0.000	0.000
3	0.000	0.000
4	1.000	1.000

A.2 ESTIMATED PARAMETERS OF THE MIDAS-AR MODEL

Table A.3 Estimated Parameters of MIDAS-AR Model

Parameter	Estimated Value
β_0	39.043
ϕ_1	0.999
β_1	-0.202
β_2	-0.214
$\nu_1 = (\nu_{11}, \nu_{12})$	(-0.033, 1.193)
$\nu_2 = (\nu_{21}, \nu_{22})$	(2.603, 1.224)

Note: ν_{11} denotes the first Exponential Almon parameter for the DXY index variable.
 ν_{12} denotes the second Exponential Almon parameter for the DXY index variable.
 ν_{21} denotes the first Exponential Almon parameter for the OIL price variable.
 ν_{22} denotes the second Exponential Almon parameter for the OIL price variable.

Table A.4 MIDAS-AR Lag Weights Based on the Estimated Exponential Almon Polynomial

k	DXY weight $w_{1,k}(\nu_1)$	OIL weight $w_{2,k}(\nu_2)$
1	0.000	0.000
2	0.000	0.000
3	0.000	0.000
4	0.000	0.000
5	0.000	0.000
6	0.000	0.000
7	0.000	0.000
8	0.000	0.000
9	0.000	0.000
10	0.000	0.000
11	0.000	0.000
12	1.000	1.000

A.3 ESTIMATED PARAMETERS OF THE MIDAS-AR-THRESHOLD MODEL

Table A.5 Estimated Parameters of the MIDAS-AR-Threshold Model for Regime 1

Parameter	Estimated Value
$\beta_{0,1}$	0.268
ϕ_1	1.109
$\beta_{1,1}$	0.035
$\beta_{2,1}$	-0.093
$\nu_{1,1} = (\nu_{11}, \nu_{12})$	(4.007, -0.230)
$\nu_{2,1} = (\nu_{21}, \nu_{22})$	(0.363, 5.785)

where $\nu_{1,1}$ denotes the vector of Exponential Almon parameters for the DXY variable in Regime 1.

$\nu_{2,1}$ denotes the vector of Exponential Almon parameters for the OIL variable in Regime 1.

Table A.6 Estimated Parameters of the MIDAS-AR-Threshold Model for Regime 2

Parameter	Estimated Value
$\beta_{0,2}$	0.869
ϕ_2	0.984
$\beta_{1,2}$	0.434
$\beta_{2,2}$	-0.183
$\nu_{1,2} = (\nu_{11}, \nu_{12})$	(1.040, 0.002)
$\nu_{2,2} = (\nu_{21}, \nu_{22})$	(1.867, -0.79)

where $\nu_{1,2}$ denotes the vector of Exponential Almon parameters for the DXY variable in Regime 2.

$\nu_{2,2}$ denotes the vector of Exponential Almon parameters for the OIL variable in Regime 2.

Table A.7 MIDAS-AR-Threshold Lag Weights Based on the Estimated Exponential Almon Polynomial for Regime 1

k	DXY weight $w_{1,k}(\nu_{1,1})$	OIL weight $w_{2,k}(\nu_{2,1})$
1	0.000	0.000
2	0.000	0.000
3	0.000	0.000
4	0.000	0.000
5	0.002	0.000
6	0.012	0.000
7	0.052	0.000
8	0.143	0.000
9	0.248	0.000
10	0.273	0.000
11	0.189	0.000
12	0.083	1.000

Table A.8 MIDAS-AR-Threshold Lag Weights Based on the Estimated Exponential Almon Polynomial for Regime 2

k	DXY weight $w_{1,k}(\nu_{1,2})$	OIL weight $w_{2,k}(\nu_{2,2})$
1	0.000	0.000
2	0.000	0.000
3	0.000	0.000
4	0.000	0.001
5	0.000	0.003
6	0.001	0.009
7	0.003	0.025
8	0.009	0.057
9	0.026	0.112
10	0.076	0.189
11	0.224	0.272
12	0.660	0.334

A.4 ESTIMATED PARAMETERS OF THE MIDAS-AR-HMM MODEL

Table A.9 Estimated Parameters of the MIDAS-AR-HMM Model for Regime 1

Parameter	Estimated Value
$\beta_{0,1}$	0.008
ϕ_1	0.023
$\beta_{1,1}$	-0.077
$\beta_{2,1}$	0.254
$\nu_{1,1} = (\nu_{11}, \nu_{12})$	(-0.610, -0.040)
$\nu_{2,1} = (\nu_{21}, \nu_{22})$	(2.010, -1.2085)

where $\nu_{1,1}$ denotes the vector of Exponential Almon parameters for the DXY variable in Regime 1.

$\nu_{2,1}$ denotes the vector of Exponential Almon parameters for the OIL variable in Regime 1.

Table A.10 Estimated Parameters of the MIDAS-AR-HMM Model for Regime 2

Parameter	Estimated Value
$\beta_{0,2}$	-0.010
ϕ_2	-0.491
$\beta_{1,2}$	-6.124
$\beta_{2,2}$	-1.183
$\nu_{1,2} = (\nu_{11}, \nu_{12})$	(0.431, -0.223)
$\nu_{2,2} = (\nu_{21}, \nu_{22})$	(-0.657, 0.129)

where $\nu_{1,2}$ denotes the vector of Exponential Almon parameters for the DXY variable in Regime 2.

$\nu_{2,2}$ denotes the vector of Exponential Almon parameters for the OIL variable in Regime 2.

Table A.11 MIDAS-AR-HMM Lag Weights Based on the Estimated Exponential Almon Polynomial for Regime 1

k	DXY weight $w_{1,k}(\nu_{1,1})$	OIL weight $w_{2,k}(\nu_{2,1})$
1	0.530	0.272
2	0.277	0.606
3	0.133	0.120
4	0.059	0.002

Table A.12 MIDAS-AR-HMM Lag Weights Based on the Estimated Exponential Almon Polynomial for Regime 2

k	DXY weight $w_{1,k}(\nu_{1,2})$	OIL weight $w_{2,k}(\nu_{2,2})$
1	0.271	0.411
2	0.333	0.238
3	0.263	0.178
4	0.133	0.172

A.5 ESTIMATED PARAMETERS OF THE MIDAS-AR-HMM-SOM MODEL

Table A.13 Estimated Parameters of the MIDAS-AR-HMM-SOM Model for Regime 1

Parameter	Estimated Value
$\beta_{0,1}$	0.060
ϕ_1	-0.288
$\beta_{1,1}$	-1.619
$\beta_{2,1}$	-0.392
$\nu_{1,1} = (\nu_{11}, \nu_{12})$	(0.149, 0.000)
$\nu_{2,1} = (\nu_{21}, \nu_{22})$	(0.145, 0.000)

where $\nu_{1,1}$ denotes the vector of Exponential Almon parameters for the DXY variable in Regime 1.

$\nu_{2,1}$ denotes the vector of Exponential Almon parameters for the OIL variable in Regime 1.

Table A.14 Estimated Parameters of the MIDAS-AR-HMM-SOM Model for Regime 2

Parameter	Estimated Value
$\beta_{0,2}$	-0.018
ϕ_2	-0.330
$\beta_{1,2}$	-0.047
$\beta_{2,2}$	0.375
$\nu_{1,2} = (\nu_{11}, \nu_{12})$	(0.002, -0.314)
$\nu_{2,2} = (\nu_{21}, \nu_{22})$	(0.102, -0.039)

where $\nu_{1,2}$ denotes the vector of Exponential Almon parameters for the DXY variable in Regime 2.

$\nu_{2,2}$ denotes the vector of Exponential Almon parameters for the OIL variable in Regime 2.

Table A.15 MIDAS-AR-HMM-SOM Lag Weights Based on the Estimated Exponential Almon Polynomial for Regime 1

k	DXY weight $w_{1,k}(\nu_{1,1})$	OIL weight $w_{2,k}(\nu_{2,1})$
1	0.197	0.209
2	0.229	0.234
3	0.266	0.263
4	0.308	0.295

Table A.16 MIDAS-AR-HMM-SOM Lag Weights Based on the Estimated Exponential Almon Polynomial for Regime 2

k	DXY weight $w_{1,k}(\nu_{1,2})$	OIL weight $w_{2,k}(\nu_{2,2})$
1	0.277	0.246
2	0.269	0.262
3	0.245	0.258
4	0.210	0.235

APPENDIX B

PYTHON CODE

This chapter presents the Python code developed for the MIDAS-AR-HMM-SOM model proposed in this thesis. The implementation provides the computational framework for model construction, parameter estimation, regime identification, and forecasting analysis.

```

1  # Import required libraries
2  import time
3  import numpy as np
4  import pandas as pd
5  from hmmlearn.hmm import GaussianHMM
6  from scipy.optimize import minimize
7  from sklearn.metrics import mean_squared_error
8  from sklearn.preprocessing import StandardScaler
9  from statsmodels.stats.diagnostic import acorr_ljungbox
10
11 # Set random seed
12 SEED = 4
13 np.random.seed(SEED)
14
15 # Input data
16 GOLD = [input data gold]
17 DXY = [input data DXY]
18 OIL = [input data OIL]
19
20 # Create date indices
21 gold_dates = pd.date_range("2001-01-31", periods=len(GOLD), freq="M")
22 weekly_dates = pd.date_range("2001-01-05", periods=len(DXY), freq="W-FRI")
23
24 gold_s = pd.Series(GOLD, index=gold_dates, dtype=float)
25 dxy_s = pd.Series(DXY, index=weekly_dates, dtype=float)
26 oil_s = pd.Series(OIL, index=weekly_dates, dtype=float)
27
28 # Compute logarithmic returns
29 gold_ret = np.log(gold_s).diff().dropna()
30 dxy_ret = np.log(dxy_s).diff().dropna()
31 oil_ret = np.log(oil_s).diff().dropna()

```

```

32
33 # Construct HMM-SOM features
34 vol = gold_ret.rolling(3).std()
35 lag_ret = gold_ret.shift(1)
36
37 df_feat = pd.DataFrame(
38     {
39         "ret": gold_ret,
40         "vol": vol,
41         "lag_ret": lag_ret,
42     }
43 ).dropna()
44
45 # Define Self-Organizing Map
46 class SOM:
47
48     def __init__(self, k, dim, seed):
49         np.random.seed(seed)
50         self.w = np.random.randn(k, dim)
51
52     def bmu(self, x):
53         return np.argmax(np.linalg.norm(self.w - x, axis=1))
54
55     def fit(self, X):
56         for t in range(1000):
57             x = X[t % len(X)]
58             b = self.bmu(x)
59             self.w[b] += 0.1 * (x - self.w[b])
60         return self
61
62     def transform(self, X):
63         return np.array([self.bmu(x) for x in X])
64
65 # Standardize features and apply SOM
66 scaler = StandardScaler()
67 X = scaler.fit_transform(df_feat[["ret", "vol", "lag_ret"]])

```

```

68
69 som = SOM(4, X.shape[1], SEED).fit(X)
70 df_feat["som"] = som.transform(X)
71
72 # Fit Hidden Markov Model
73 hmm_data = df_feat[["ret", "vol", "lag_ret", "som"]].values
74
75 hmm = GaussianHMM(
76     n_components=2,
77     n_iter=200,
78     random_state=SEED,
79 )
80
81 hmm.fit(hmm_data)
82 df_feat["regime"] = hmm.predict(hmm_data)
83
84
85 # Stable Exponential Almon weighting function
86 def almon(K, t1, t2):
87
88     j = np.arange(K)
89     z = t1 * j + t2 * j * j
90     z = z - np.max(z)
91
92     w = np.exp(z)
93     s = np.sum(w)
94
95     if (s == 0) or np.isnan(s) or np.isnan(w).any() or np.isinf(w).any():
96         return np.ones(K) / K
97
98     return w / s
99
100 # Extract last K weekly observations
101 def get_last_k(series, date, K):
102
103     vals = series.loc[series.index <= date].tail(K).values

```

```

104
105     if len(vals) == K:
106         return vals[:-1]
107
108     return None
109
110
111 # Build MIDAS dataset
112 K = 4
113 rows = []
114
115 idx = df_feat.index
116
117 for t in range(1, len(idx)):
118
119     dt = idx[t]
120
121     dxy_l = get_last_k(dxy_ret, dt, K)
122     oil_l = get_last_k(oil_ret, dt, K)
123
124     if dxy_l is None or oil_l is None:
125         continue
126
127     row = {
128         "date": dt,
129         "y": df_feat.loc[dt, "ret"],
130         "yl": df_feat.loc[idx[t - 1], "ret"],
131         "regime": df_feat.loc[dt, "regime"],
132     }
133
134     for j in range(K):
135         row[f"d{j}"] = dxy_l[j]
136         row[f"o{j}"] = oil_l[j]
137
138     rows.append(row)
139

```

```

140 df = pd.DataFrame(rows).set_index("date")
141
142 # Train-test split
143 TRAIN_RATIO = 0.70
144
145 split_idx = int(len(df) * TRAIN_RATIO)
146
147 train = df.iloc[:split_idx]
148 test = df.iloc[split_idx:]
149
150 # MIDAS-AR prediction function
151 def predict(p, Xd, Xo, yl):
152
153     a, p1, bd, t1, t2, bo, s1, s2 = p
154
155     wd = almon(K, t1, t2)
156     wo = almon(K, s1, s2)
157
158     y = a + p1 * yl + bd * (Xd @ wd) + bo * (Xo @ wo)
159
160     if np.isnan(y).any():
161         return np.zeros_like(yl)
162
163     return y
164
165 # Objective function
166 def loss(p, Xd, Xo, yl, y):
167
168     yhat = predict(p, Xd, Xo, yl)
169
170     penalty = 0.01 * (
171         p[3] ** 2
172         + p[4] ** 2
173         + p[6] ** 2
174         + p[7] ** 2
175     )

```

```

176
177     return np.sum((y - yhat) ** 2) + penalty
178
179 # Estimate MIDAS-AR parameters
180 def fit(df_sub):
181
182     Xd = df_sub[[f"d{i}" for i in range(K)]].values
183     Xo = df_sub[[f"o{i}" for i in range(K)]].values
184
185     y = df_sub["y"].values
186     yl = df_sub["yl"].values
187
188     init = [0, 0.2, 0.1, -0.1, -0.01, 0.1, -0.1, -0.01]
189
190     bounds = [
191         (-1, 1),
192         (-1, 1),
193         (-5, 5),
194         (-1, 1),
195         (-1, 0),
196         (-5, 5),
197         (-1, 1),
198         (-1, 0),
199     ]
200
201     res = minimize(
202         loss,
203         init,
204         args=(Xd, Xo, yl, y),
205         bounds=bounds,
206         method="L-BFGS-B",
207     )
208
209     return res.x
210
211 p0 = fit(train[train.regime == 0])

```

```
212 p1 = fit(train[train.regime == 1])
213
214 # Remaining functions (predict_df, return_to_price, calc_metrics,
215 # calc_aic_bic, and the print section) should follow the same
216 # indentation style shown above.
```
