ps5

April 10, 2023

1. Helen Xiao, Alex Cai, Iñaki Arango
2. Total: 12 (Theory: 4, Computational: 8)
3. numpy help online
4. I have not shared my code with anyone and have not used anyone else's code.

[18]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import yfinance as yfin
import math
import matplotlib_inline.backend_inline
import statsmodels.api as sm
from pandas_datareader import data as pdr
import plotly.graph_objects as go

matplotlib_inline.backend_inline.set_matplotlib_formats('pdf', 'png')

plt.rcParams['savefig.dpi'] = 75

plt.rcParams['figure.autolayout'] = False
plt.rcParams['figure.figsize'] = 10, 6
plt.rcParams['axes.labelsize'] = 18
plt.rcParams['axes.titlesize'] = 20
plt.rcParams['font.size'] = 16
plt.rcParams['lines.linewidth'] = 2.0
plt.rcParams['lines.markersize'] = 8
plt.rcParams['legend.fontsize'] = 14

plt.rcParams['text.usetex'] = True
plt.rcParams['font.family'] = "serif"
plt.rcParams['font.serif'] = "cm"
```

Problem 1: Lagrange Multipliers

PROBLEM 1

(1) $\mathcal{L}(x,y) = 4x - 3y + \lambda(x^2+y^2-100)$

$\frac{d\mathcal{L}}{dx} = 4 + 2\lambda x = 0$

$\frac{d\mathcal{L}}{dy} = -3 + 2\lambda y = 0$

$\Big\} -\frac{2}{x} = \frac{3}{2y} \rightarrow -4y = 3x \rightarrow y = -\frac{3x}{4} \rightarrow y = -\frac{3}{4}(\pm 8) \rightarrow \underline{\underline{y = \pm 6}}$

$\frac{d\mathcal{L}}{d\lambda} = x^2+y^2-100 = 0 \rightarrow x^2 + \left(-\frac{3x}{4}\right)^2 = 100 \rightarrow x^2 + \frac{9x^2}{16} = 100 \rightarrow 25x^2 = 1600 \rightarrow x^2 = 64 \rightarrow \underline{\underline{x = \pm 8}}$

$\boxed{\max f(x,y) = 50}$

(2) $(x-y)^2 \geq 0$

$x^2+y^2 \geq 2xy$

$x^2+2xy+y^2 \geq 4xy$

$(x+y)^2 \geq 4xy$

$x+y \geq 2\sqrt{xy}$

$\frac{x+y}{2} \geq \sqrt{xy}$

*equality holds when $x=y$

EXTRA CREDIT

NEED TO PROVE: $\frac{\sum_{i=1}^{n} x_i}{n} \geq \left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}}$

$\log\left(\frac{x_1+\ldots+x_n}{n}\right) \geq \frac{\log(x_1)+\ldots+\log(x_n)}{n}$ * $\log(x)$ = concave by Jenson's inequality

$\log\left(\frac{x_1+\ldots+x_n}{n}\right) \geq \frac{\log(x_1 \cdot \ldots \cdot x_n)}{n}$

$\log\left(\frac{\sum_{i=1}^{n} x_i}{n}\right) \geq \log\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}}$

$\frac{\sum_{i=1}^{n} x_i}{n} \geq \left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}}$ ✓

(3) $\frac{x+y}{2} \geq \sqrt{xy}$

$\frac{1}{2} \geq \sqrt{xy}$ *$x+y=1$

$\frac{1}{4} \geq xy \rightarrow \boxed{\max xy = \frac{1}{4}}$

$\mathcal{L}(x,y) = xy + \lambda(x+y-1)$

$\frac{d\mathcal{L}}{dx} = y + \lambda = 0$

$\frac{d\mathcal{L}}{dy} = x + \lambda = 0$

$\Big\} x = y \rightarrow y = \frac{1}{2}$

$\frac{d\mathcal{L}}{d\lambda} = x+y-1 = 0 \rightarrow x+x=1 \rightarrow \underline{\underline{x = \frac{1}{2}}}$

$\boxed{\max xy \text{ st. } x+y=1 = \frac{1}{4}}$

(4) $\max v'\Sigma v$ st. $\|v\|^2 = 1$ * $\|v\| = 1 \rightarrow \|v\|^2 = 1$

$\mathcal{L}(v) = v'\Sigma v + \lambda(\|v\|^2 - 1)$

$\frac{d\mathcal{L}}{dv} = 2\Sigma v - \lambda(2v) = 0 \rightarrow \Sigma v = \lambda v$

$\frac{d\mathcal{L}}{d\lambda} = \|v\|^2 - 1 = 0 \rightarrow \|v\|^2 = 1$

$f(v) = v'\Sigma v$

$= v'(\lambda v)$

$= \lambda v'v$

$= \lambda \|v\|^2$

$= \lambda$

λ maximizes $v'\Sigma v$ st. $\|v\|=1$ and is the largest eigenvector of Σ

[ ]:
```
# 1.5 Eigenvector Simulation
```

[15]:
```
# 2.1 Returns

stock_ids = ('GOOG', 'AAPL', 'TSLA')
yfin.pdr_override()
stock_data = pdr.get_data_yahoo(stock_ids, datetime(2013, 2, 1),
                                datetime(2023, 2, 1))["Adj Close"]
net_returns = 100 * (stock_data - stock_data.shift(1)) / stock_data.shift(1)
net_returns = net_returns.dropna()
```

```
net_returns.head()

mu = net_returns.mean(axis=0) * 252.0
mu
```

```
[**********100%**********]  3 of 3 completed
```

[15]:
```
AAPL    27.561301
GOOG    20.136823
TSLA    58.824375
dtype: float64
```

[14]:
```
# 2.1 Covariance Matrix

cov = net_returns.cov() * 252.0
cov
```

[14]:

	AAPL	GOOG	TSLA
AAPL	827.316090	456.246504	629.886152
GOOG	456.246504	739.551200	567.702140
TSLA	629.886152	567.702140	3333.130784

[21]:
```
# 2.1 Efficient Frontier

n_pf = 10000 #number of portfolios in our simulation
sample_pf = 0.5 * np.random.normal(size=(len(stock_ids), n_pf))
mu_pf = np.array(mu) @ sample_pf
var_pf = (sample_pf.T @ np.array(cov) * sample_pf.T).sum(-1)
ef_flag = np.zeros_like(mu_pf)
for i in range(n_pf):
  if all(mu_pf[var_pf < var_pf[i]] < mu_pf[i]):
    ef_flag[i] = 1

fig = go.Figure()
fig.add_trace(
    go.Scatter(mode='markers',
               x=np.sqrt(var_pf),
               y=mu_pf,
               marker=dict(color='black', size=1),
               showlegend=False))
fig.add_trace(
    go.Scatter(mode='markers',
               x=np.sqrt(var_pf[np.nonzero(ef_flag)]),
               y=mu_pf[np.nonzero(ef_flag)],
               marker=dict(color='blue', size=2),
               name='Optimal'))
fig.add_trace(
```

```
    go.Scatter(mode='markers',
               x=np.sqrt(np.diag(np.array(cov))),
               y=np.array(mu),
               marker=dict(color='crimson', size=8),
               name='Individual'))
fig.update_layout(xaxis_title="Volatility",
                  yaxis_title="Mean",
                  height=600,
                  width=700)
fig.update_xaxes(range=[-3, 50])
fig.update_yaxes(range=[-50, 50])
fig.show()
```

Problem 2

(2) $\min w'\Sigma_t w$ s.t. $w'1 = 1$

$\mathcal{L}(w) = w'\Sigma_t w + \lambda(w'1 - 1)$

$\frac{d\mathcal{L}}{dw} = 2\Sigma w - \lambda 1 = 0 \rightarrow 2\Sigma w = \lambda 1 \rightarrow w = \Sigma^{-1}\frac{\lambda}{2}1 \rightarrow \boxed{w = \frac{\Sigma^{-1}1}{1'\Sigma^{-1}1}}$

$\frac{d\mathcal{L}}{d\lambda} = w'1 - 1 = 0 \rightarrow 1'w = 1 \rightarrow 1'\Sigma^{-1}\frac{\lambda}{2}1 = 1 \rightarrow \boxed{\lambda = \frac{2}{1'\Sigma^{-1}1}}$

(3) $\Sigma_t = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_n^2 \end{bmatrix}$

$w = \frac{\Sigma_t^{-1}1}{1'\Sigma_t^{-1}1} = \frac{\begin{bmatrix} \frac{1}{\sigma_1^2} \\ \vdots \\ \frac{1}{\sigma_n^2} \end{bmatrix}}{\sum_{i=1}^{n}\frac{1}{\sigma_i}}$

portfolios allocate more capital towards assets with less variance since the smaller σ_i^2, the larger $\frac{1}{\sigma_i^2}$ which increases its weight in the portfolio

(4) $E[V_{t+1}|F_t] = \frac{1}{T_t}\mu_t'\Sigma_t^{-1}\mu_t$

$Var(V_{t+1}|F_t) = \frac{1}{T_t^2}\mu_t'\Sigma_t^{-1}\mu_t$

[43]:
```
# 3.1 High Dimensional Covariance Matrix Estimation: p = 3
mean = np.zeros(3)
cov = np.identity(3)

simulations = np.random.multivariate_normal(mean, cov, size=1000)
cov_matrix = np.cov(simulations.T)
```

```
np.linalg.eigvals(cov_matrix)
```

[43]:
```
array([1.04524036, 0.9751096 , 0.92040428])
```

From these results, we can see that the eigenvalues are close to 1 which indicates that the covariance matrix estimation is close to the actual covariance matrix.

[44]:
```
# 3.2 High Dimensional Covariance Matrix Estimation: p = 1000
mean_1000 = np.zeros(1000)
cov_1000 = np.identity(1000)

simulations_1000 = np.random.multivariate_normal(mean_1000, cov_1000, size=1000)
cov_matrix_1000 = np.cov(simulations_1000.T)

np.linalg.eigvals(cov_matrix_1000)
```

[44]:
```
array([3.94494091e+00, 3.93090471e+00, 3.87563123e+00, 3.85302316e+00,
       3.82204586e+00, 3.79984969e+00, 3.77975195e+00, 3.75072185e+00,
       3.71921038e+00, 3.71188306e+00, 3.68740956e+00, 3.63171109e+00,
       3.61727426e+00, 3.60335051e+00, 3.59528283e+00, 3.58005248e+00,
       3.55901931e+00, 3.54524410e+00, 3.52752132e+00, 3.49815353e+00,
       3.50139319e+00, 3.47487181e+00, 3.46225987e+00, 3.41682085e+00,
       3.42839819e+00, 3.44155015e+00, 3.40503990e+00, 3.36765166e+00,
       3.35273108e+00, 3.33099643e+00, 3.31552641e+00, 3.30509282e+00,
       3.29158782e+00, 3.26976459e+00, 3.27999503e+00, 3.24884225e+00,
       3.24218739e+00, 3.23074518e+00, 3.21642689e+00, 3.20212342e+00,
       3.18834055e+00, 3.17867435e+00, 3.17142281e+00, 3.16033408e+00,
       3.14971184e+00, 3.09987459e+00, 3.11591896e+00, 3.13385611e+00,
       3.13076058e+00, 3.08008944e+00, 3.07261256e+00, 3.05713331e+00,
       3.05916664e+00, 3.04377969e+00, 3.02740730e+00, 3.02400257e+00,
       3.00522376e+00, 3.00294798e+00, 2.99132679e+00, 2.97681355e+00,
       2.96205566e+00, 2.95250997e+00, 2.94343204e+00, 2.92833463e+00,
       2.93743621e+00, 2.90976547e+00, 2.90230402e+00, 2.89792344e+00,
       2.88884250e+00, 2.87449407e+00, 2.86153310e+00, 2.85772584e+00,
       2.84470884e+00, 2.83986485e+00, 2.82583413e+00, 2.81839037e+00,
       2.80512072e+00, 2.80035235e+00, 2.76153950e+00, 2.78413630e+00,
       2.77382492e+00, 2.61111638e+00, 2.74416689e+00, 2.62521269e+00,
       2.70030671e+00, 2.78005966e+00, 2.71333524e+00, 2.73696195e+00,
       2.68305992e+00, 2.64520269e+00, 2.63242940e+00, 2.63512574e+00,
       2.65857016e+00, 2.67565611e+00, 2.66353951e+00, 2.66738038e+00,
       2.72378783e+00, 2.72492265e+00, 2.58878119e+00, 2.59255370e+00,
       2.60576847e+00, 2.60033018e+00, 2.57534574e+00, 2.56606893e+00,
       2.56127245e+00, 2.54838685e+00, 2.54084918e+00, 2.51429240e+00,
       2.52333437e+00, 2.52075105e+00, 2.49934922e+00, 2.49587818e+00,
       2.20345386e+00, 2.21295858e+00, 2.47764181e+00, 2.48162662e+00,
       2.46486575e+00, 2.45227451e+00, 2.45621042e+00, 2.21971161e+00,
```

2.22503767e+00, 2.46291746e+00, 2.43659948e+00, 2.43099243e+00,
2.48318628e+00, 2.23496345e+00, 2.23770061e+00, 2.41880943e+00,
2.25154161e+00, 2.41039617e+00, 2.25585134e+00, 2.40200428e+00,
2.32576048e+00, 2.31653535e+00, 2.39529321e+00, 2.28828316e+00,
2.39864949e+00, 2.38105519e+00, 2.30146565e+00, 2.28061184e+00,
2.34057107e+00, 2.37176218e+00, 2.36076259e+00, 2.29740313e+00,
2.27371933e+00, 2.36555959e+00, 2.34844049e+00, 2.26865424e+00,
2.27003103e+00, 2.35379883e+00, 2.37582654e+00, 1.97239466e+00,
2.19494032e+00, 1.98204954e+00, 1.98360075e+00, 2.18207175e+00,
2.17715053e+00, 1.99302210e+00, 2.00146686e+00, 2.16799059e+00,
2.16437568e+00, 2.00765686e+00, 2.15775523e+00, 2.01269589e+00,
2.15208492e+00, 2.01957627e+00, 2.03450044e+00, 2.02394589e+00,
2.04159664e+00, 2.04727901e+00, 2.05535897e+00, 2.14258417e+00,
2.06966092e+00, 2.12352057e+00, 2.11209528e+00, 2.12976599e+00,
2.13881808e+00, 2.10192107e+00, 2.07442635e+00, 2.09830202e+00,
2.09021154e+00, 2.08158372e+00, 2.13312547e+00, 2.00983703e+00,
2.07893787e+00, 2.09256848e+00, 2.11457031e+00, 1.81580034e+00,
1.82258537e+00, 1.82964640e+00, 1.83587105e+00, 1.84009818e+00,
1.84426615e+00, 1.85140855e+00, 1.84882712e+00, 1.85684787e+00,
1.86589745e+00, 1.89084392e+00, 1.89898689e+00, 1.87884487e+00,
1.96024416e+00, 1.90643487e+00, 1.87070614e+00, 1.95204677e+00,
1.92142454e+00, 1.91747196e+00, 1.88305389e+00, 1.91327901e+00,
1.94573689e+00, 1.93199639e+00, 1.94026833e+00, 1.93989998e+00,
1.88514285e+00, 1.69813439e+00, 1.70875164e+00, 1.71242784e+00,
1.71826551e+00, 1.72664477e+00, 1.72823828e+00, 1.73696601e+00,
1.74084819e+00, 1.75007420e+00, 1.75609464e+00, 1.76617788e+00,
1.76022500e+00, 1.77267961e+00, 1.78434162e+00, 1.80798928e+00,
1.79555748e+00, 1.78829696e+00, 1.79002366e+00, 1.80142107e+00,
1.80289764e+00, 1.75776452e+00, 1.60787961e+00, 1.61228802e+00,
1.61577000e+00, 1.62141790e+00, 1.62827135e+00, 1.63139734e+00,
1.63505812e+00, 1.63656886e+00, 1.65068649e+00, 1.65646476e+00,
1.65893984e+00, 1.66504032e+00, 1.67508301e+00, 1.68090871e+00,
1.68949805e+00, 1.70070912e+00, 1.69367901e+00, 1.64908916e+00,
1.52478925e+00, 1.52960756e+00, 1.53477181e+00, 1.53836507e+00,
1.54560716e+00, 1.54957029e+00, 1.55898885e+00, 1.56413211e+00,
1.58864784e+00, 1.58649080e+00, 1.59809039e+00, 1.56809778e+00,
1.57837898e+00, 1.57222468e+00, 1.60633731e+00, 1.56997091e+00,
1.44920771e+00, 1.45314902e+00, 1.45606154e+00, 1.46490905e+00,
1.46924315e+00, 1.47489054e+00, 1.48236634e+00, 1.51059687e+00,
1.51543063e+00, 1.48617131e+00, 1.50467711e+00, 1.49872313e+00,
1.48961099e+00, 1.49465603e+00, 1.49079447e+00, 1.49971457e+00,
1.37366342e+00, 1.37638587e+00, 1.38286879e+00, 1.38622098e+00,
1.39058898e+00, 1.39855712e+00, 1.40338226e+00, 1.40708782e+00,
1.41241669e+00, 1.41865622e+00, 1.42431248e+00, 1.43102088e+00,
1.44032563e+00, 1.44228039e+00, 1.43714127e+00, 1.42679479e+00,
1.29764924e+00, 1.30174915e+00, 1.30952730e+00, 1.31212905e+00,
1.31504082e+00, 1.32234619e+00, 1.32716446e+00, 1.33138673e+00,

1.33545581e+00, 1.34443259e+00, 1.35351125e+00, 1.35526301e+00,
1.34030202e+00, 1.36731280e+00, 1.36138676e+00, 1.36500531e+00,
1.21562084e+00, 1.21751718e+00, 1.22329095e+00, 1.23058580e+00,
1.23236674e+00, 1.23528616e+00, 1.23830128e+00, 1.25356336e+00,
1.26364772e+00, 1.24643506e+00, 1.26617565e+00, 1.24378703e+00,
1.27539469e+00, 1.24966334e+00, 1.29129770e+00, 1.28979521e+00,
1.28165012e+00, 1.28376718e+00, 1.27848037e+00, 1.27174050e+00,
1.12509166e+00, 1.12813971e+00, 1.13105635e+00, 1.13564165e+00,
1.14654935e+00, 1.14297359e+00, 1.15278403e+00, 1.15763110e+00,
1.21084606e+00, 1.20405409e+00, 1.19730164e+00, 1.18642562e+00,
1.18400799e+00, 1.21367531e+00, 1.20114978e+00, 1.16895124e+00,
1.16505156e+00, 1.18023646e+00, 1.16379564e+00, 1.17510095e+00,
1.14173540e+00, 1.17191103e+00, 1.19186450e+00, 1.12023501e+00,
1.11601102e+00, 1.10986135e+00, 1.10630032e+00, 1.10019791e+00,
1.09597141e+00, 1.09349002e+00, 1.09031817e+00, 1.08535224e+00,
1.07781534e+00, 1.08292239e+00, 1.07372546e+00, 1.07687954e+00,
1.06823350e+00, 1.06037113e+00, 1.05884252e+00, 1.05500800e+00,
1.05232046e+00, 1.03171782e+00, 1.04748647e+00, 1.04570238e+00,
1.03991489e+00, 1.04299526e+00, 1.02697029e+00, 1.00074223e+00,
1.02839952e+00, 1.02016276e+00, 1.01006993e+00, 1.00486292e+00,
1.01321117e+00, 1.01314887e+00, 1.02175365e+00, 9.98282354e-01,
9.92257096e-01, 9.95237285e-01, 9.87829080e-01, 9.80917730e-01,
9.76895518e-01, 9.85768202e-01, 9.71081417e-01, 9.85077336e-01,
9.62428713e-01, 9.67076174e-01, 9.64020245e-01, 9.54460660e-01,
9.48434855e-01, 9.53758987e-01, 9.43530637e-01, 9.51065545e-01,
9.36661953e-01, 9.33089890e-01, 9.26459570e-01, 9.19434811e-01,
9.32365456e-01, 9.28041773e-01, 7.83796857e-01, 9.15240019e-01,
9.15379158e-01, 8.98683310e-01, 9.04522967e-01, 9.06679574e-01,
9.11366406e-01, 7.86827184e-01, 8.95557799e-01, 7.89384080e-01,
8.92814603e-01, 7.90209508e-01, 8.86340961e-01, 7.91308654e-01,
8.90843668e-01, 7.95918205e-01, 8.82380501e-01, 8.78836404e-01,
7.97914730e-01, 8.75922957e-01, 8.72811910e-01, 8.01674586e-01,
8.05178164e-01, 8.66942217e-01, 8.08314398e-01, 8.11198667e-01,
8.68491549e-01, 8.62488682e-01, 8.56912372e-01, 8.52139623e-01,
8.16813058e-01, 8.47032231e-01, 8.59549277e-01, 8.19631889e-01,
8.23285003e-01, 8.25048157e-01, 8.26512464e-01, 8.36000867e-01,
8.34667065e-01, 8.39785829e-01, 8.15082523e-01, 8.42775781e-01,
8.49455236e-01, 7.61686143e-01, 7.65429591e-01, 7.69303679e-01,
7.74695572e-01, 7.75722028e-01, 7.79443354e-01, 7.18331237e-01,
7.22987227e-01, 7.56716947e-01, 7.54792669e-01, 7.51900835e-01,
7.31175966e-01, 7.28865668e-01, 7.34359988e-01, 7.43973172e-01,
7.48836337e-01, 7.42522709e-01, 7.36624982e-01, 7.26675340e-01,
7.26757462e-01, 7.12888301e-01, 7.15984054e-01, 7.10774184e-01,
7.09055248e-01, 7.02077150e-01, 7.05453614e-01, 6.98086454e-01,
6.94136810e-01, 6.95463371e-01, 6.87446054e-01, 6.87394090e-01,
6.23577003e-01, 6.27755587e-01, 6.82487275e-01, 6.79239753e-01,
6.32208635e-01, 6.75701741e-01, 6.73553926e-01, 6.59405205e-01,

6.54452273e-01, 6.38572762e-01, 6.49699677e-01, 6.69145559e-01,
6.43750293e-01, 6.66581244e-01, 6.46706962e-01, 6.67912321e-01,
6.51381818e-01, 6.40789825e-01, 6.44294689e-01, 6.64383812e-01,
6.57730123e-01, 6.35060162e-01, 6.32743352e-01, 6.18973495e-01,
6.18191322e-01, 6.13253691e-01, 6.11511817e-01, 6.10261624e-01,
6.08162260e-01, 6.01765168e-01, 5.96662269e-01, 5.94915289e-01,
6.04084912e-01, 6.04518192e-01, 5.91465562e-01, 4.85041110e-01,
4.87868851e-01, 5.86056714e-01, 5.82811742e-01, 5.80187026e-01,
5.81020073e-01, 4.89065393e-01, 5.77798858e-01, 5.75483826e-01,
4.91648749e-01, 5.64167415e-01, 5.62608347e-01, 5.55831585e-01,
5.73941936e-01, 5.53986625e-01, 5.70616846e-01, 5.66698269e-01,
4.93079905e-01, 5.72086418e-01, 5.68151597e-01, 5.51323460e-01,
4.98483481e-01, 4.96031080e-01, 5.46307269e-01, 5.43688686e-01,
5.40819509e-01, 5.02396317e-01, 5.08804610e-01, 5.06128322e-01,
5.12081250e-01, 5.04396183e-01, 5.35522113e-01, 5.36317260e-01,
5.24137667e-01, 5.27508683e-01, 5.30715603e-01, 5.29585928e-01,
5.20065403e-01, 5.21595366e-01, 5.33136549e-01, 5.15898039e-01,
5.15981670e-01, 5.16919648e-01, 4.72794435e-01, 4.74114957e-01,
4.75698260e-01, 4.81825603e-01, 4.83409064e-01, 4.80244576e-01,
4.34092999e-01, 4.35976023e-01, 4.47730134e-01, 4.53620229e-01,
4.51231959e-01, 4.36953217e-01, 4.42692732e-01, 4.40530623e-01,
4.67028159e-01, 4.60365601e-01, 4.65898865e-01, 4.57575645e-01,
4.69525600e-01, 4.43436119e-01, 4.58121562e-01, 4.62218064e-01,
4.29986991e-01, 4.32809659e-01, 3.58232076e-01, 4.25525843e-01,
4.24239330e-01, 3.60658272e-01, 4.22180477e-01, 3.59771801e-01,
3.63498433e-01, 4.20871703e-01, 4.16667168e-01, 3.64590305e-01,
3.67488146e-01, 3.85960995e-01, 3.89523966e-01, 3.83927490e-01,
3.81441141e-01, 4.13268204e-01, 3.79109612e-01, 4.19069415e-01,
3.93938173e-01, 3.73705691e-01, 3.94758027e-01, 4.08576842e-01,
4.06868258e-01, 4.10608541e-01, 3.71959595e-01, 3.77339963e-01,
3.97873510e-01, 4.04088477e-01, 4.11949664e-01, 4.02759017e-01,
3.70327866e-01, 4.01023494e-01, 3.77098524e-01, 4.00104861e-01,
3.96558707e-01, 3.52981264e-01, 3.54723606e-01, 3.46453777e-01,
3.49217486e-01, 3.50203802e-01, 3.48269149e-01, 3.37526875e-01,
3.32645899e-01, 3.43082695e-01, 3.40957280e-01, 3.41343806e-01,
3.35877123e-01, 3.31560787e-01, 2.76091844e-01, 2.78218903e-01,
3.28869435e-01, 3.24255214e-01, 3.27341354e-01, 3.27108610e-01,
3.22165178e-01, 3.19276185e-01, 2.80151280e-01, 2.80885034e-01,
2.83485976e-01, 3.15055260e-01, 3.16546824e-01, 3.17368788e-01,
3.12456218e-01, 2.85697765e-01, 3.09868847e-01, 3.06625453e-01,
3.04791047e-01, 3.00791668e-01, 2.87842811e-01, 2.94642126e-01,
3.08935189e-01, 2.99373288e-01, 2.90619548e-01, 2.91366656e-01,
2.96708218e-01, 2.88751260e-01, 3.18228278e-01, 2.88914020e-01,
3.05955518e-01, 2.54908789e-01, 2.55895106e-01, 2.64198892e-01,
2.69914175e-01, 2.60239153e-01, 2.57618916e-01, 2.74259160e-01,
2.61409228e-01, 2.67142477e-01, 2.72450696e-01, 2.71282174e-01,
2.73574702e-01, 2.59404984e-01, 2.51067405e-01, 2.50028906e-01,

2.48199251e-01, 2.47705681e-01, 2.56645637e-01, 2.42659525e-01,
2.41971801e-01, 2.44162713e-01, 2.34715081e-01, 2.37141110e-01,
2.38019112e-01, 2.34068238e-01, 2.32441890e-01, 2.31117426e-01,
2.02587870e-01, 2.02959264e-01, 2.29298225e-01, 2.26255209e-01,
2.21692363e-01, 2.20298006e-01, 2.23767002e-01, 2.07906360e-01,
2.09436865e-01, 2.05087809e-01, 2.14679558e-01, 2.18203706e-01,
2.24540092e-01, 2.08426067e-01, 2.29723079e-01, 2.12526453e-01,
2.11416614e-01, 2.16144701e-01, 2.17089189e-01, 2.00142435e-01,
1.99823811e-01, 1.94632226e-01, 1.96008319e-01, 1.97808960e-01,
1.38760917e-01, 1.93544697e-01, 1.92563771e-01, 1.89834813e-01,
1.88910023e-01, 1.97677288e-01, 1.91051168e-01, 1.86180178e-01,
1.84932782e-01, 1.87731777e-01, 1.82746043e-01, 1.81504027e-01,
1.41832621e-01, 1.44028634e-01, 1.78385850e-01, 1.79096243e-01,
1.45157245e-01, 1.46392409e-01, 1.75479878e-01, 1.74201257e-01,
1.70572874e-01, 1.47584532e-01, 1.77804418e-01, 1.73355852e-01,
1.76561317e-01, 1.69277609e-01, 1.49320937e-01, 1.49887698e-01,
1.50412919e-01, 1.67537220e-01, 1.63358642e-01, 1.56254937e-01,
1.64938799e-01, 1.62002959e-01, 1.58718130e-01, 1.63051508e-01,
1.60330909e-01, 1.66740132e-01, 1.53419047e-01, 1.53990526e-01,
1.54578530e-01, 1.53237729e-01, 1.60665913e-01, 1.34385330e-01,
1.35431995e-01, 1.36329699e-01, 1.39093761e-01, 1.40653307e-01,
1.32796138e-01, 1.31286024e-01, 1.29913385e-01, 1.28908705e-01,
1.27357009e-01, 1.27043226e-01, 1.25196454e-01, 1.24899169e-01,
1.22591658e-01, 1.21703655e-01, 1.20585768e-01, 1.19664392e-01,
1.18579346e-01, 1.11429240e-01, 1.14290331e-01, 1.13494613e-01,
1.16129967e-01, 1.19389216e-01, 1.17239194e-01, 1.16417959e-01,
1.10104791e-01, 1.09087026e-01, 1.06660318e-01, 1.07937686e-01,
1.03497948e-01, 1.08233461e-01, 1.02247818e-01, 1.04261855e-01,
1.04048246e-01, 1.00257192e-01, 9.89390782e-02, 9.87509896e-02,
9.71181491e-02, 9.58148506e-02, 1.00894551e-01, 9.35382362e-02,
9.51856116e-02, 9.48513815e-02, 9.18401320e-02, 9.24992625e-02,
8.82609063e-02, 8.97384935e-02, 9.04664542e-02, 8.68009926e-02,
8.54688343e-02, 5.30665760e-02, 5.32955189e-02, 8.46634322e-02,
8.88693939e-02, 5.42245638e-02, 5.49541453e-02, 5.56813113e-02,
8.21848853e-02, 8.35952259e-02, 8.37578553e-02, 5.62444398e-02,
8.13738578e-02, 7.94749435e-02, 7.74483383e-02, 7.43935453e-02,
5.72285547e-02, 5.77282913e-02, 5.85302100e-02, 7.19545269e-02,
5.92540667e-02, 7.86114540e-02, 6.98237734e-02, 7.27111787e-02,
6.92402355e-02, 7.05879802e-02, 7.60206775e-02, 6.05660576e-02,
6.79622184e-02, 7.60049877e-02, 6.27211034e-02, 6.36013639e-02,
6.58777652e-02, 6.51075831e-02, 6.48276759e-02, 7.91609287e-02,
6.16202948e-02, 6.10243664e-02, 6.72433551e-02, 6.20548926e-02,
8.03774282e-02, 6.68126072e-02, 4.95880589e-02, 5.02762050e-02,
5.14160544e-02, 4.85237160e-02, 3.70435036e-02, 4.31010849e-02,
4.26511281e-02, 4.65322164e-02, 4.74266609e-02, 4.72884475e-02,
4.18316951e-02, 4.55519942e-02, 4.42556186e-02, 4.81632505e-02,
3.97119961e-02, 3.79156766e-02, 3.91905291e-02, 4.59303104e-02,

```
4.10399658e-02, 4.44057621e-02, 4.07518895e-02, 3.85857600e-02,
3.58659517e-02, 3.54501107e-02, 3.35627178e-02, 3.30267293e-02,
3.17578091e-02, 3.13901748e-02, 3.46458893e-02, 3.05648639e-02,
3.44418621e-02, 3.04321104e-02, 2.95955627e-02, 2.80519624e-02,
2.88824861e-02, 3.39298137e-02, 2.76546878e-02, 2.53665921e-02,
2.67561091e-02, 3.49506071e-02, 2.48805985e-02, 2.60426426e-02,
2.87070945e-02, 2.40859065e-02, 2.62017989e-02, 2.36760063e-02,
2.32792960e-02, 2.23377636e-02, 2.18113996e-02, 2.73068466e-02,
2.11569692e-02, 2.04938011e-02, 2.09151862e-02, 1.81661966e-02,
1.87959133e-02, 1.76144643e-02, 1.99318572e-02, 1.72367598e-02,
1.95874363e-02, 1.97154121e-02, 1.68040595e-02, 1.58466499e-02,
1.54159191e-02, 1.66334963e-02, 2.25694339e-02, 1.22411371e-02,
1.31559585e-02, 1.43045281e-02, 1.38442432e-02, 1.51652419e-02,
1.44727045e-02, 1.36367615e-02, 1.24905361e-02, 1.27170584e-02,
1.13090745e-02, 1.28831752e-02, 1.10591363e-02, 8.87793289e-03,
9.67184845e-03, 1.01289333e-02, 1.03209266e-02, 9.36421508e-03,
8.51619605e-03, 8.33113092e-03, 1.07936013e-02, 1.08593283e-02,
7.96053089e-03, 9.78143674e-03, 7.59966472e-03, 6.77646378e-03,
5.43514460e-03, 5.66752981e-03, 5.56312440e-03, 6.66210591e-03,
6.35184103e-03, 7.11858013e-03, 7.38056621e-03, 4.98582500e-03,
4.10508921e-03, 6.10112274e-03, 3.79375037e-03, 4.62347732e-03,
3.56133270e-03, 4.71393255e-03, 2.68006816e-03, 4.78954131e-03,
2.38245948e-03, 2.32365108e-03, 3.42493742e-03, 3.30571394e-03,
1.77839009e-03, 1.84868150e-03, 3.00969872e-03, 7.31693118e-03,
3.09763242e-03, 1.32926916e-03, 1.28243050e-03, 2.04829828e-03,
1.48489406e-03, 2.92897977e-03, 1.02785783e-03, 8.51597647e-04,
2.01272447e-03, 1.51463931e-03, 7.45894136e-04, 8.68451599e-05,
1.46534052e-04, 4.05422900e-05, 1.67484130e-05, 7.25013606e-06,
1.30463507e-16, 1.12519643e-03, 3.33245888e-04, 1.93178221e-04,
2.11754987e-04, 3.45834373e-07, 4.91976115e-04, 6.41146832e-04,
5.34480085e-04, 4.34895380e-04, 5.76473820e-05, 2.01689484e-04,
4.22126536e-04, 6.45656727e-04, 3.57047298e-04, 4.32755508e-03])
```

In contrast to 3.1, the covariance matrix estimation deviates quite a lot from the original covariance matrix as we can see from most of the eigenvalues hovering between 3 and 4 instead of 1. This shows how it can be hard to estimate covariance matrix from data when the number of assets is of the same order of magnitude or larger than the number of time points we have data for.

PROBLEM 4

(1) $\text{cov}(B_t^2, B_s) = E[B_t^2 B_s] - E[B_t^2]E[B_s]$

$= E[B_t^2 B_s]$

$= E[(B_t - B_s + B_s)^2 B_s]$

$= E[(B_t - B_s)^2 B_s + 2(B_t - B_s)B_s^2 + B_s^3]$

$= E[(B_t - B_s)^2]E[B_s] + 2E[B_t - B_s]E[B_s^2] + E[B_s^3]$

$= 0$

(2) $X = \sum_{i=1}^{n} a_i B(t_i)$ is a linear combination of normal random variables hence X is Normally distr

$E[X] = \sum_{i=1}^{n} a_i E[B(t_i)] = \sum_{i=1}^{n} a_i (0) = \boxed{0}$

$\text{Var}(X) = \sum_{i,j=1}^{n} a_i a_j \text{Cov}(B(t_i), B(t_j))$

$= \sum_{i=1}^{n} a_i^2 \text{Var}(B(t_i)) + \sum_{i \neq j} a_i a_j \text{Cov}(B(t_i), B(t_j))$

$= \sum_{i=1}^{n} a_i^2 t_i + 2\sum_{i<j} a_i a_j (t_i \wedge t_j)$

(3) $Z_t = \frac{B_{ct}}{\sqrt{c}}$

① $Z_0 = \frac{B_0}{\sqrt{c}} = 0$ ✓

② Z_t has stationary + independent increments

NEED TO SHOW: $Z_{t_1+s} - Z_{t_1} \overset{d}{=} Z_{t_2+s} - Z_{t_2}$

$Z_{t_1+s} - Z_{t_1} = \frac{B_{ct_1+cs}}{\sqrt{c}} - \frac{B_{ct_1}}{\sqrt{c}} = \frac{1}{\sqrt{c}}(B_{ct_1+cs} - B_{ct_1}) \sim N(0, \frac{cs}{c}) = N(0,s)$

$Z_{t_2+s} - Z_{t_2} = \frac{B_{ct_2+cs}}{\sqrt{c}} - \frac{B_{ct_2}}{\sqrt{c}} = \frac{1}{\sqrt{c}}(B_{ct_2+cs} - B_{ct_2}) \sim N(0, \frac{cs}{c}) = N(0,s)$

$\Rightarrow$ stationary increme

NEED TO SHOW: $Z_{t_1+s} - Z_{t_1} \perp Z_{t_2+s} - Z_{t_2}$

$B_{ct_1+cs} - B_{ct_1} \perp B_{ct_2+cs} - B_{ct_2} \Rightarrow \frac{B_{ct_1+cs} - B_{ct_1}}{\sqrt{c}} \perp \frac{B_{ct_2+cs} - B_{ct_2}}{\sqrt{c}} \Rightarrow Z_{t_1+s} - Z_{t_1} \perp Z_{t_2+s} - Z_{t_2} \Rightarrow$ indep

③ $Z_{t+s} - Z_s = \frac{B_{c(t+s)}}{\sqrt{c}} - \frac{B_{cs}}{\sqrt{c}} = \frac{B_{ct+cs} - B_{cs}}{\sqrt{c}}$

$B_{ct+cs} - B_{cs} \sim N(0, ct)$, $t, s > 0$

$Z_{t+s} - Z_s \sim N(0, t)$ ✓

④ $Z_t = \frac{B_{ct}}{\sqrt{c}} = \frac{1}{\sqrt{c}} B_{ct}$

$\Rightarrow B_{ct}$ has continuous sample paths so Z_t does as well

(4) $P\left(\frac{|B(t)|}{t} > \epsilon\right) = 2P\left(\frac{B(t)}{t} < -\epsilon\right)$

$= 2\Phi(-\epsilon\sqrt{t})$

$= 2(1 - \Phi(\epsilon\sqrt{t}))$ ✓

$B(t) \sim N(0,t)$

$\frac{B(t)}{t} \sim N(0, \frac{1}{t})$

$F(x) = \Phi\left(\frac{x-0}{\frac{1}{\sqrt{t}}}\right) = \Phi(\sqrt{t}\,x)$

Problem 4: Brownian Motion Computations

(5) $Z_t = a(X_t - Y_t)$

① $Z_0 = a(0-0) = 0$ ✓

② Z_t has stationary + independent increments

NEED TO SHOW: $Z_{t_1+s} - Z_{t_1} \overset{d}{=} Z_{t_2+s} - Z_{t_2}$

$Z_{t_1+s} - Z_{t_1} = a(X_{t_1+s} - Y_{t_1+s}) - a(X_{t_1} - Y_{t_1}) = a((X_{t_1+s} - X_{t_1}) - (Y_{t_1+s} - Y_{t_1})) \; ; \; X_{t_1+s} - X_{t_1} \overset{d}{=} Y_{t_1+s} - Y_{t_1} \sim N(0,s)$

$Z_{t_2+s} - Z_{t_2} = a(X_{t_2+s} - Y_{t_2+s}) - a(X_{t_2} - Y_{t_2}) = a((X_{t_2+s} - X_{t_2}) - (Y_{t_2+s} - Y_{t_2})) \; ; \; X_{t_2+s} - X_{t_2} \overset{d}{=} Y_{t_2+s} - Y_{t_2} \sim N(0,s)$

NEED TO SHOW: $Z_{t_1+s} - Z_{t_1} \perp Z_{t_2+s} - Z_{t_2}$

$X_{t_1+s} - X_{t_1} \perp X_{t_2+s} - X_{t_2}$ and $Y_{t_1+s} - Y_{t_1} \perp Y_{t_2+s} - Y_{t_2}$

$\Rightarrow (X_{t_1+s} - X_{t_1}) - (Y_{t_1+s} - Y_{t_1}) \perp (X_{t_2+s} - X_{t_2}) - (Y_{t_2+s} - Y_{t_2})$

$\Rightarrow a((X_{t_1+s} - X_{t_1}) - (Y_{t_1+s} - Y_{t_1})) \perp a((X_{t_2+s} - X_{t_2}) - (Y_{t_2+s} - Y_{t_2}))$

$\Rightarrow Z_{t_1+s} - Z_{t_1} \perp Z_{t_2+s} - Z_{t_2}$ ✓

③ $Z_{t+s} - Z_s = a((X_{t+s} - X_s) - (Y_{t+s} - Y_s))$

$X_{t+s} - X_s \sim N(0,t) \quad Y_{t+s} - Y_s \sim N(0,t)$

$(X_{t+s} - X_t) - (Y_{t+s} - Y_s) \sim N(0, 2t)$

$a((X_{t+s} - X_t) - (Y_{t+s} - Y_s)) \sim N(0,t)$

$\Rightarrow \boxed{a = \frac{\sqrt{2}}{2}}$

④ $Z_t = a(X_t - Y_t)$

$\Rightarrow X_t$ and Y_t have continuous sample paths so Z_t does as well

PROBLEM 5

(1) $E[m_{t+1}|F_t] = E[B_{t+1}^2 - (t+1)|F_t]$

$= E[(B_{t+1} - B_t + B_t)^2|F_t] - t - 1$

$= E[(B_{t+1}-B_t)^2|F_t] + 2E[B_{t+1}-B_t|F_t]E[B_t|F_t] + E[B_t^2|F_t] - t - 1$

$= t + B_t^2 - t - 1$

$= B_t^2 - t$

$= m_t$ ✓

(2) $T = \inf\{t > 0, B_t = A \text{ or } B_t = -B\}$ $\quad P(B_T = A) = \frac{B}{A+B}$

① $m_t = B_t^2 - t$ is a martingale as proved above

② $E[m_\tau] = E[m_0] = 0$ by Doob

$E[m_\tau] = E[B_\tau^2 - \tau] = 0$

$= P(B_\tau = A)A^2 + P(B_\tau = B)B^2 - E[\tau] = 0$

$= \frac{B}{A+B}A^2 + \frac{A}{A+B}B^2 - E[\tau] = 0$ by prop of martingales

$= AB - E[\tau] = 0$

$\Rightarrow E[\tau] = AB$ ✓

(3) $E[m_{t+1}|F_t] = E[B_{t+1}^3 - 3(t+1)B_{t+1}|F_t]$

$= E[(B_{t+1} - B_t + B_t)^3 - 3(t+1)(B_{t+1} - B_t + B_t)|F_t]$

$= E[(B_{t+1}-B_t)^3|F_t] + 3E[B_t|F_t]E[(B_{t+1}-B_t)^2|F_t] + 3E[B_t^2|F_t]E[B_{t+1}-B_t|F_t]$

$- 3(t+1)E[(B_{t+1}-B_t)|F_t] - 3(t+1)E[B_t|F_t]$

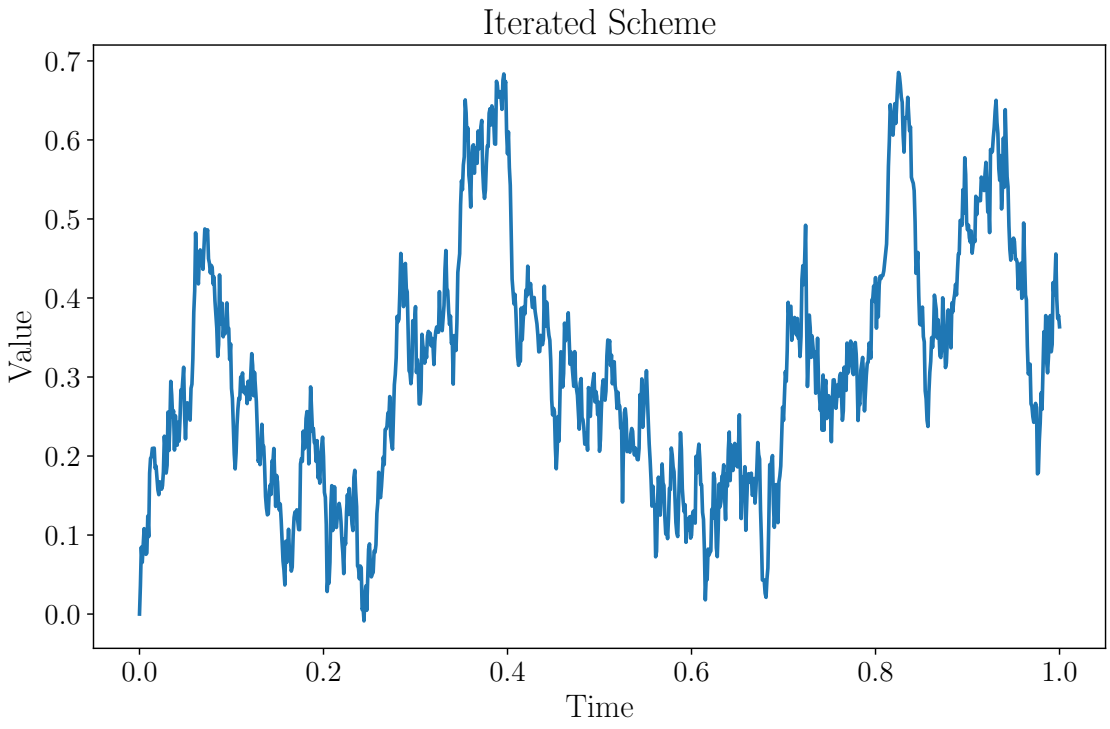
$= 3B_t(t+1-t) + B_t^3 - 3(t+1)B_t$

$= B_t^3 - 3tB_t$

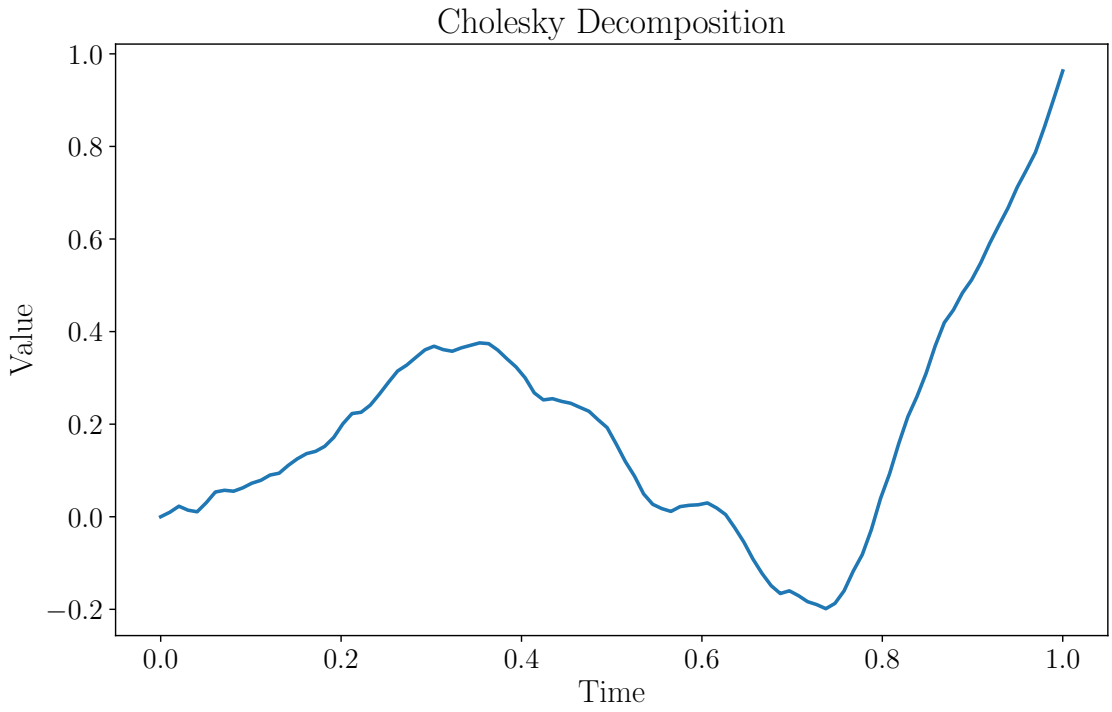
$= m_t$ ✓

Problem 5: Brownian Motion and Martingales

[62]:
```
# 6.1 Brownian Motion Simulation: Iterated Scheme

def iterated_scheme():
    n = 1000
    timepoints = np.linspace(0, 1, n + 1)
    time = timepoints[1] - timepoints[0]
    increments = np.random.normal(0, np.sqrt(time), n)

    W = np.cumsum(increments)
    W = np.concatenate(([0], W))
    plt.plot(timepoints, W)
    plt.title('Iterated Scheme')
    plt.xlabel('Time')
    plt.ylabel('Value')
    plt.show()

iterated_scheme()
```

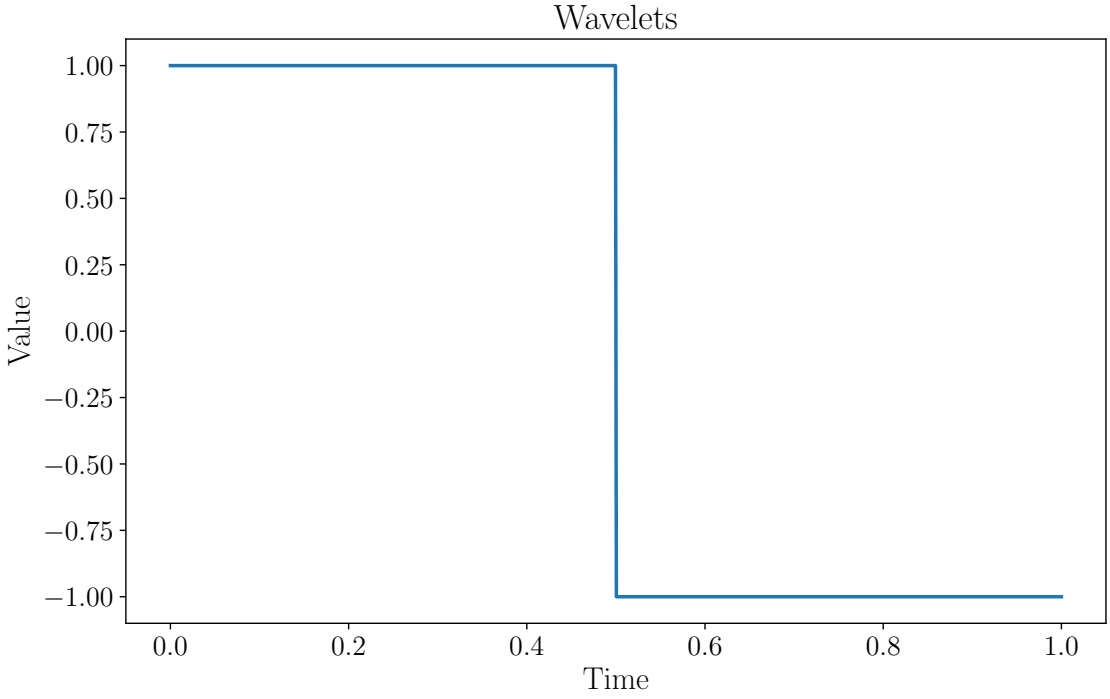


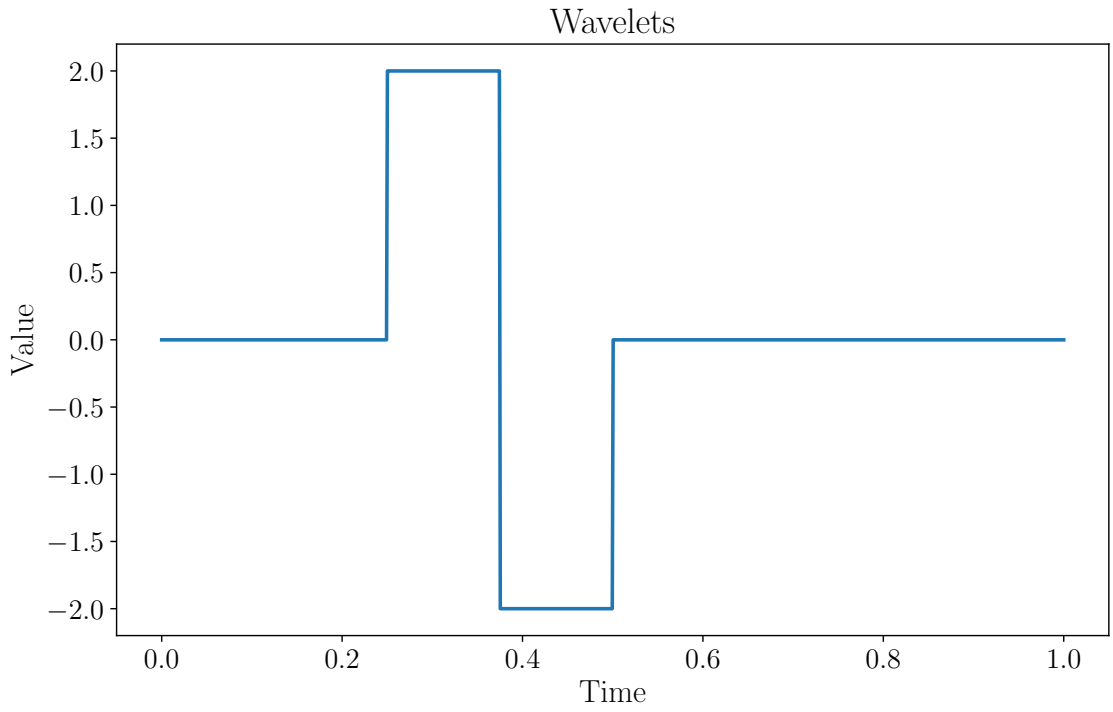
[100]:
```
# 6.2 Cholesky Decomposition

def cholesky():
    n = 100
    timepoints = np.linspace(0.0000000001, 1, n)
    time = timepoints[1] - timepoints[0]

    covariance = np.minimum.outer(timepoints, timepoints)

    decomp = np.linalg.cholesky(covariance)

    # Generate a sample path of Brownian motion
    increments = np.random.normal(0, np.sqrt(time), n)
    value = np.dot(decomp, increments)
    value = np.cumsum(value)

    plt.plot(timepoints, value)
    plt.title('Cholesky Decomposition')
    plt.xlabel('Time')
    plt.ylabel('Value')
    plt.show()

cholesky()
```

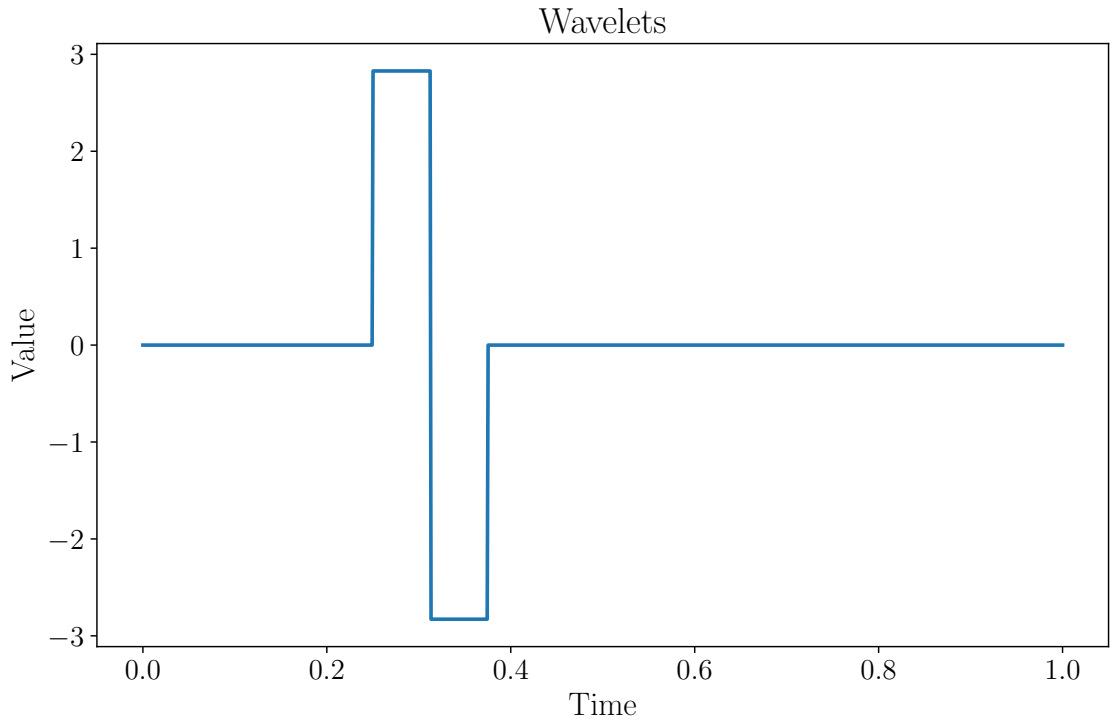


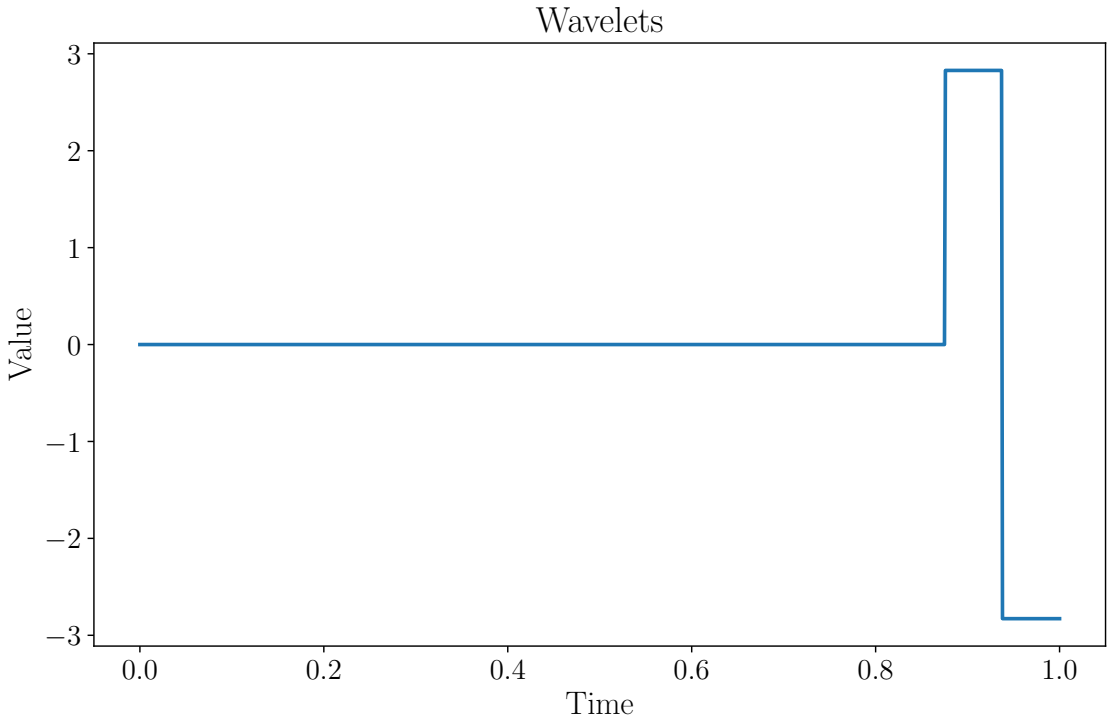
[122]:
```
# 6.3 (a)

def mother(t):
    if 0 <= t < 0.5:
        return 1
    if 0.5 < t <= 1:
        return -1
    return 0

def daughter(n, t):
    j = math.floor(math.log(n, 2))
    k = n - 2**j
    new_t = (2**j)*t - k
    x = 2**(j/2)
    return (x * mother(new_t))

def wavelets(n):
    x = 1000
    time = np.linspace (0, 1, x)
    daughters = [daughter(n, time[t]) for t in range(x)]
    plt.plot(time, daughters)
    plt.title('Wavelets')
    plt.xlabel('Time')
    plt.ylabel('Value')
```

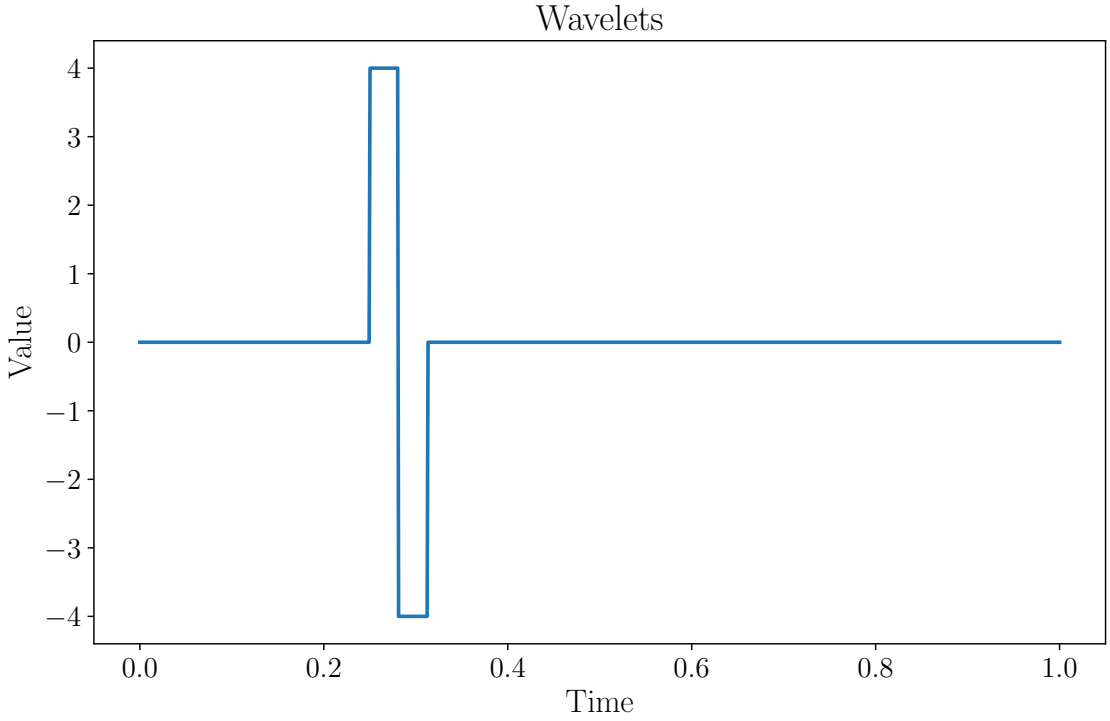
```
    plt.show()

samples = [1, 5, 10, 15, 20]
for sample in samples:
    wavelets(sample)
```



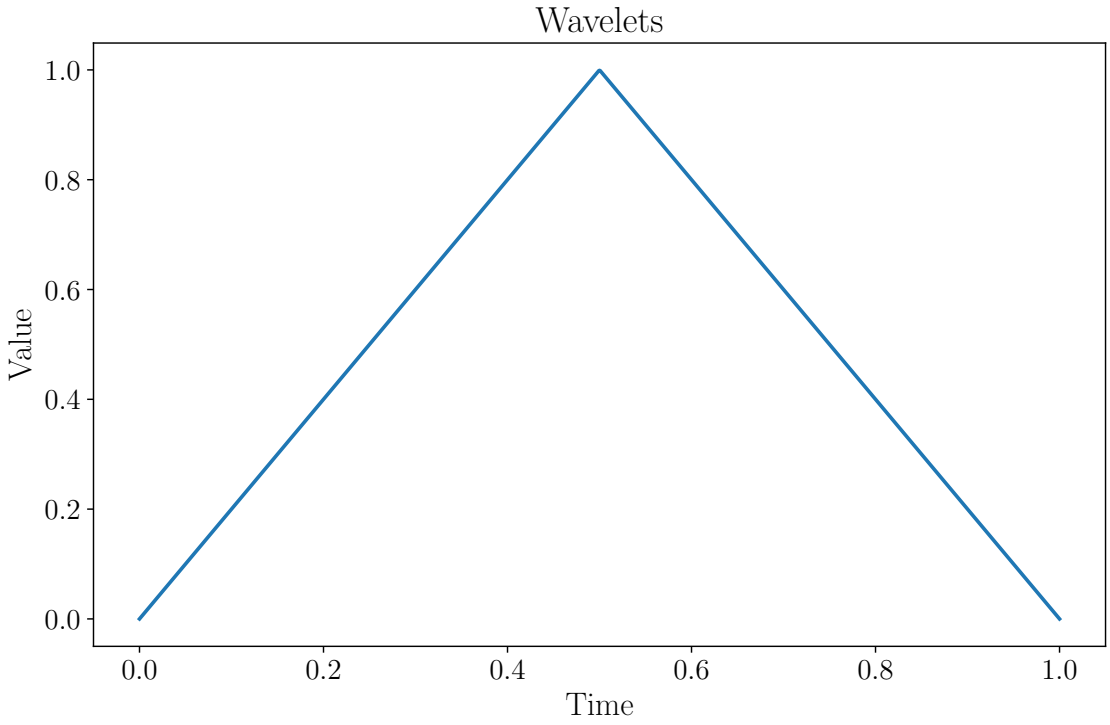

Wavelets
Value
Time
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0

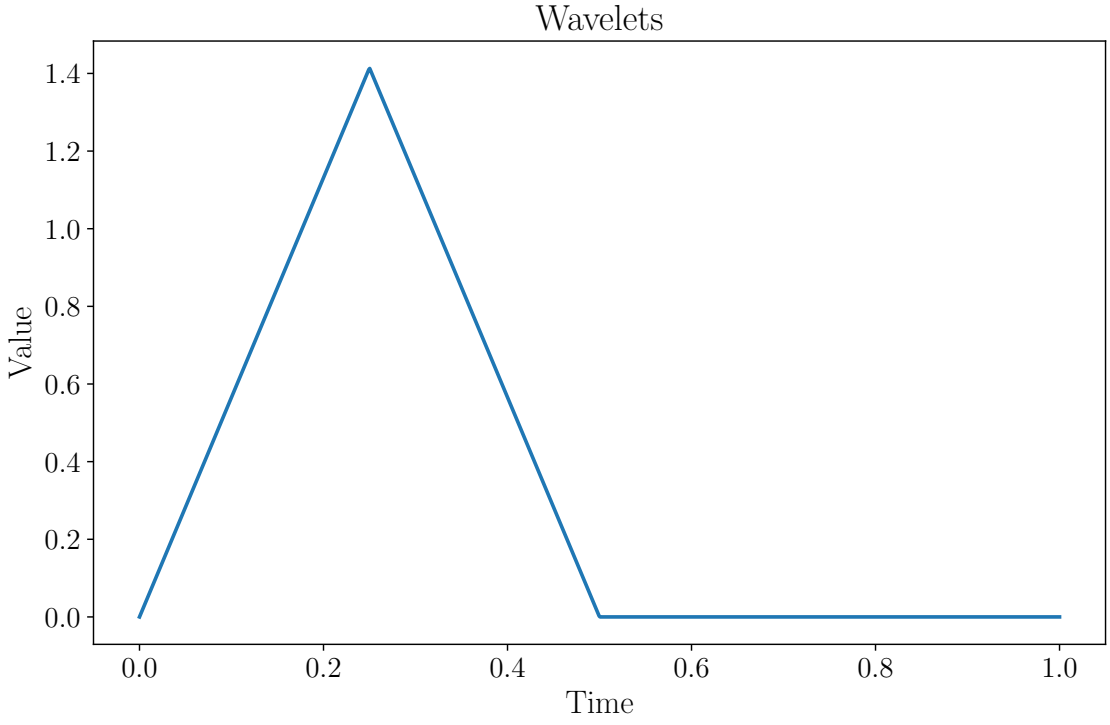

Wavelets
Value
Time
3
2
1
0
−1
−2
−3
0.0
0.2
0.4
0.6
0.8
1.0

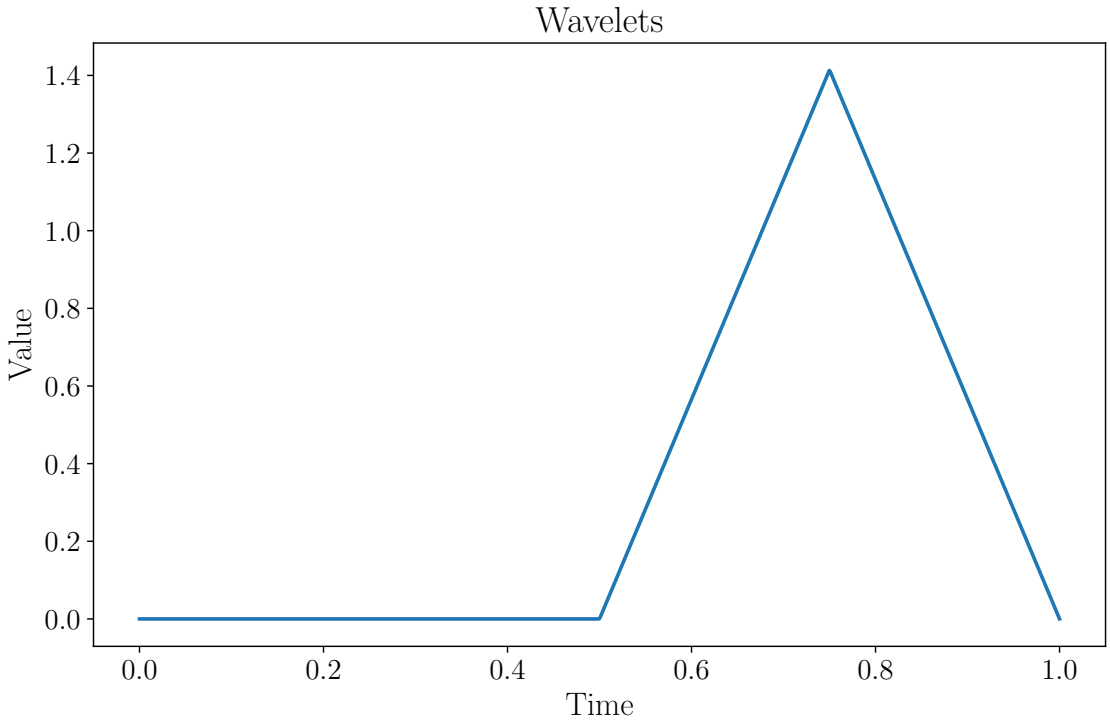

Wavelets
Value
Time
3
2
1
0
−1
−2
−3
0.0
0.2
0.4
0.6
0.8
1.0

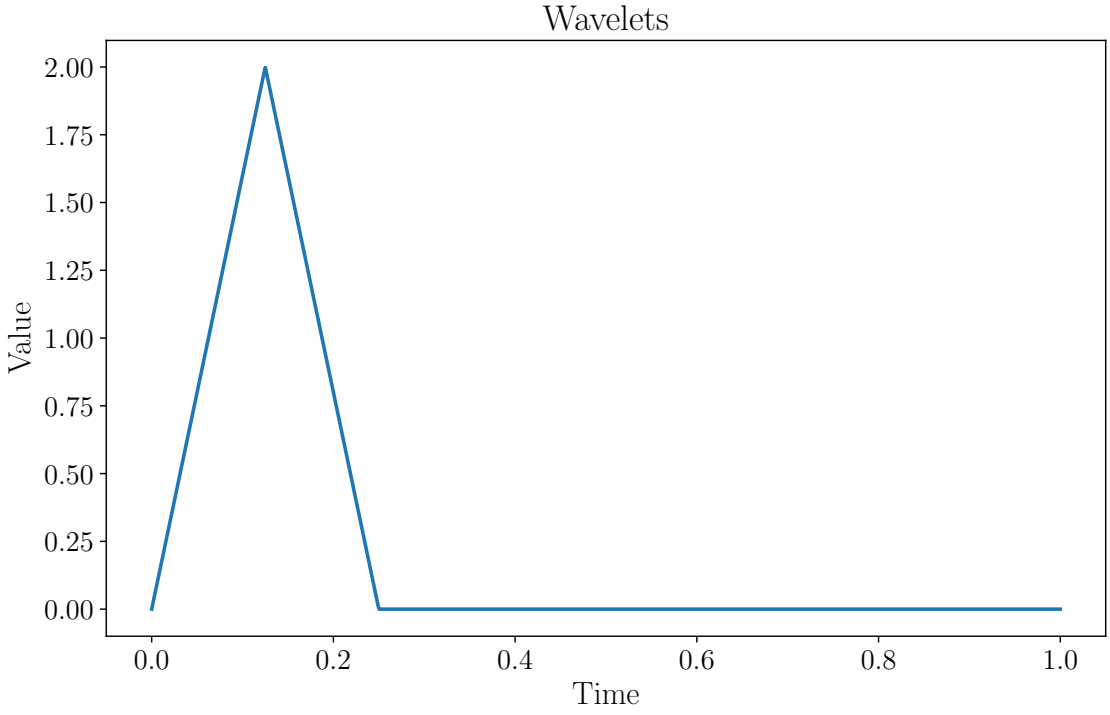

Wavelets
Value
Time
4
3
2
1
0
−1
−2
−3
−4
0.0
0.2
0.4
0.6
0.8
1.0

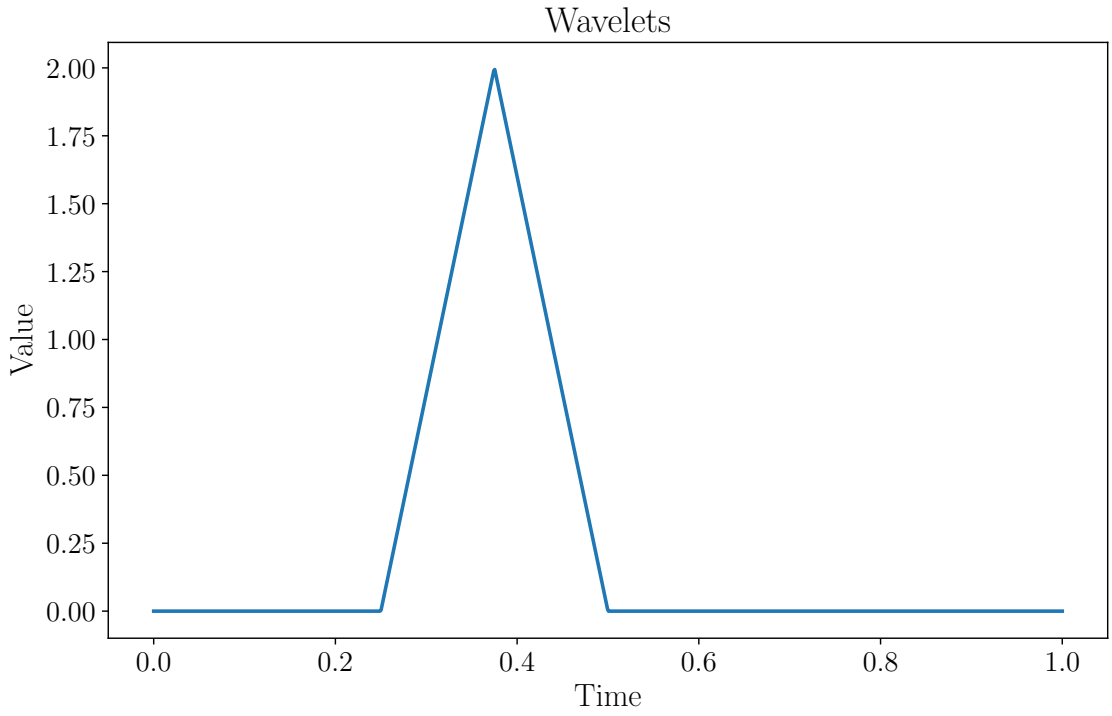
[126]:
```
# 6.3 (b)

def mother(t):
    if 0 <= t < 0.5:
        return 2*t
    if 0.5 < t <= 1:
        return 2*(1 -t)
    return 0

def daughter(n, t):
    j = math.floor(math.log(n, 2))
    k = n - 2**j
    new_t = (2**j)*t - k
    x = 2**(j/2)
    return (x * mother(new_t))

def wavelets(n):
    x = 1000
    time = np.linspace (0, 1, x)
    daughters = [daughter(n, time[t]) for t in range(x)]
    plt.plot(time, daughters)
    plt.title('Wavelets')
    plt.xlabel('Time')
    plt.ylabel('Value')
    plt.show()

for x in range(1, 33):
    wavelets(x)
```

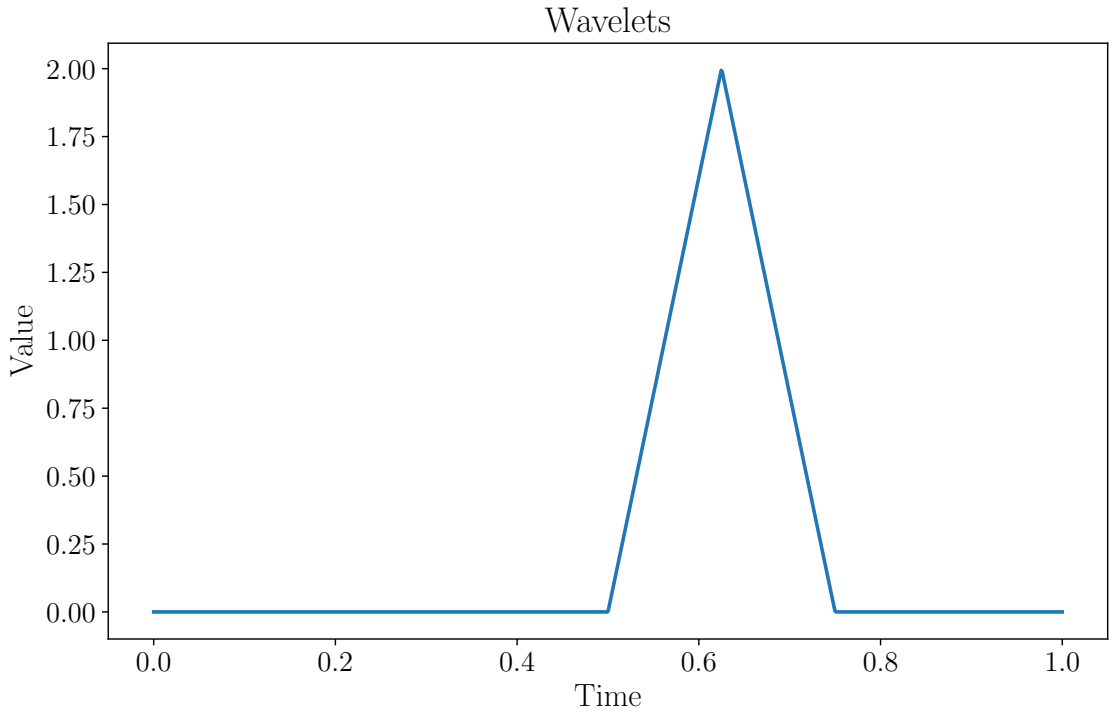

Wavelets
Value
Time
0.0
0.2
0.4
0.6
0.8
1.0

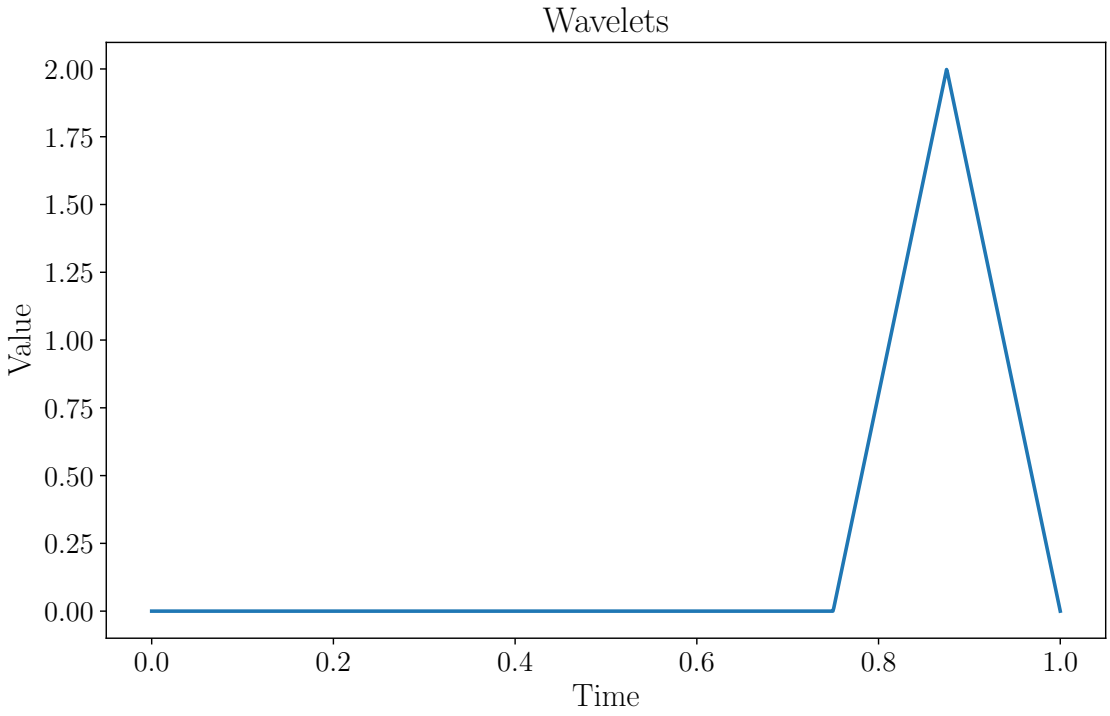

Wavelets
Value
Time
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

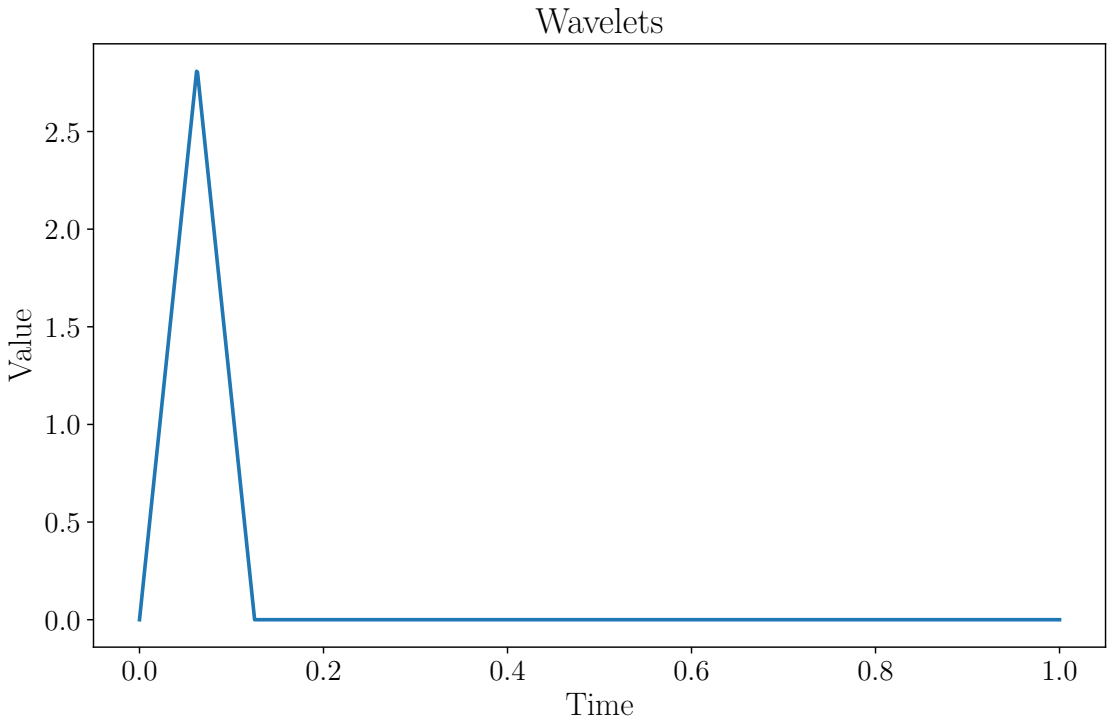

Wavelets
Value
Time
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

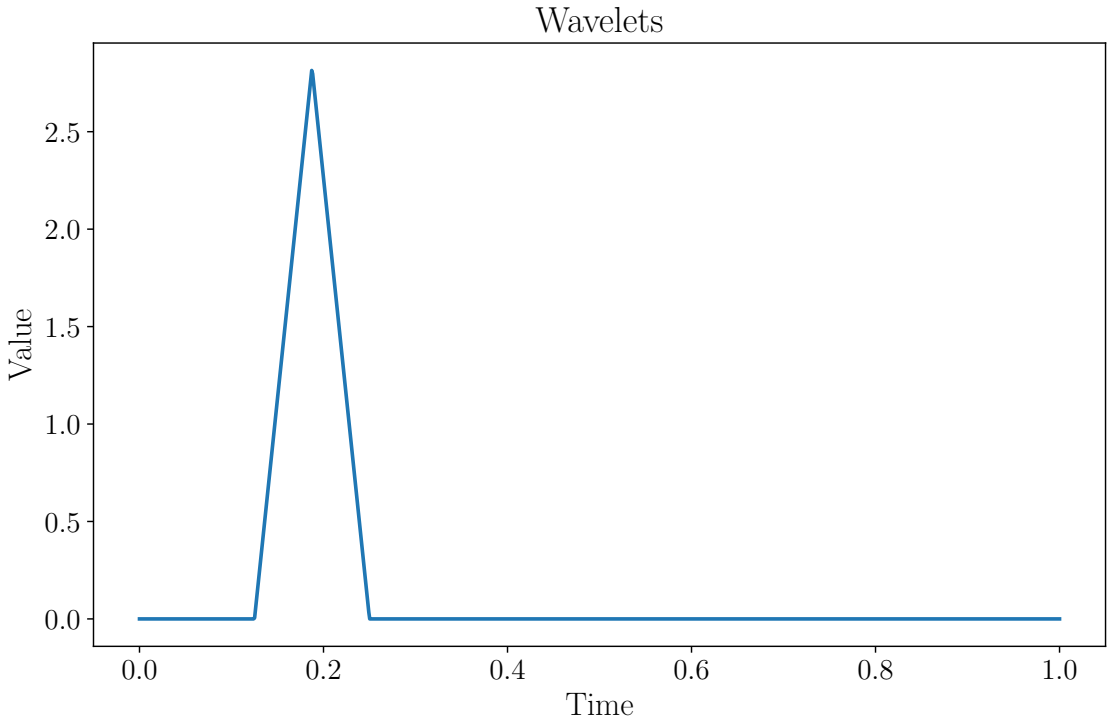

Wavelets
Value
Time
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0.0
0.2
0.4
0.6
0.8
1.0

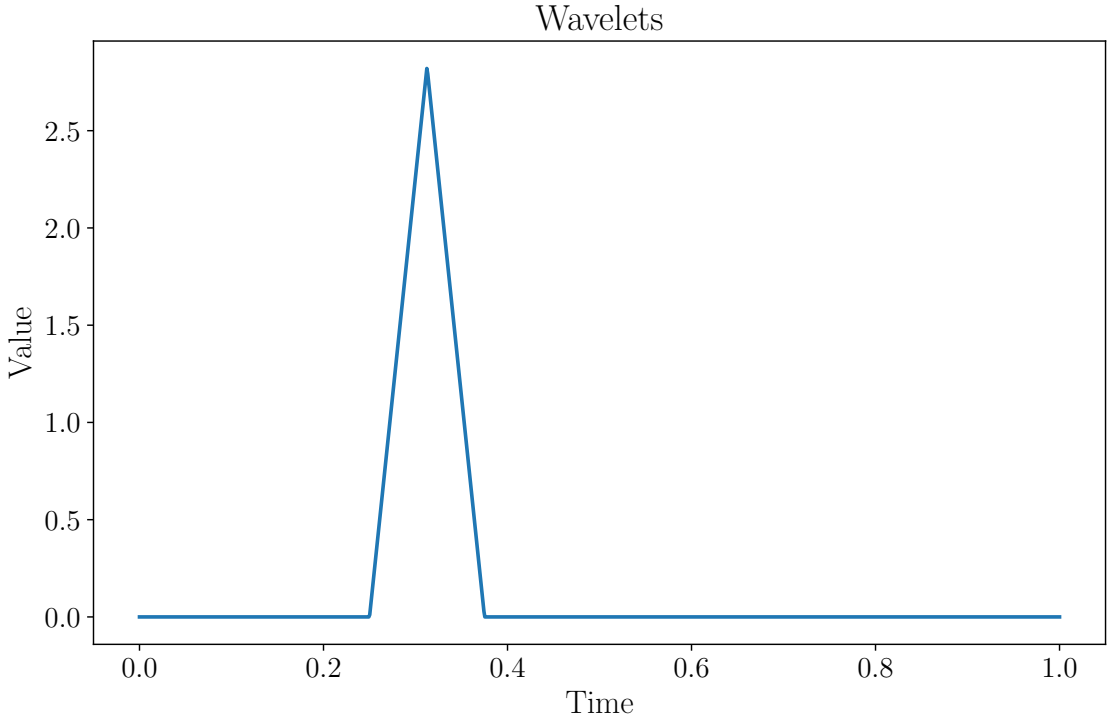

Wavelets
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Value
0.0
0.2
0.4
0.6
0.8
1.0
Time

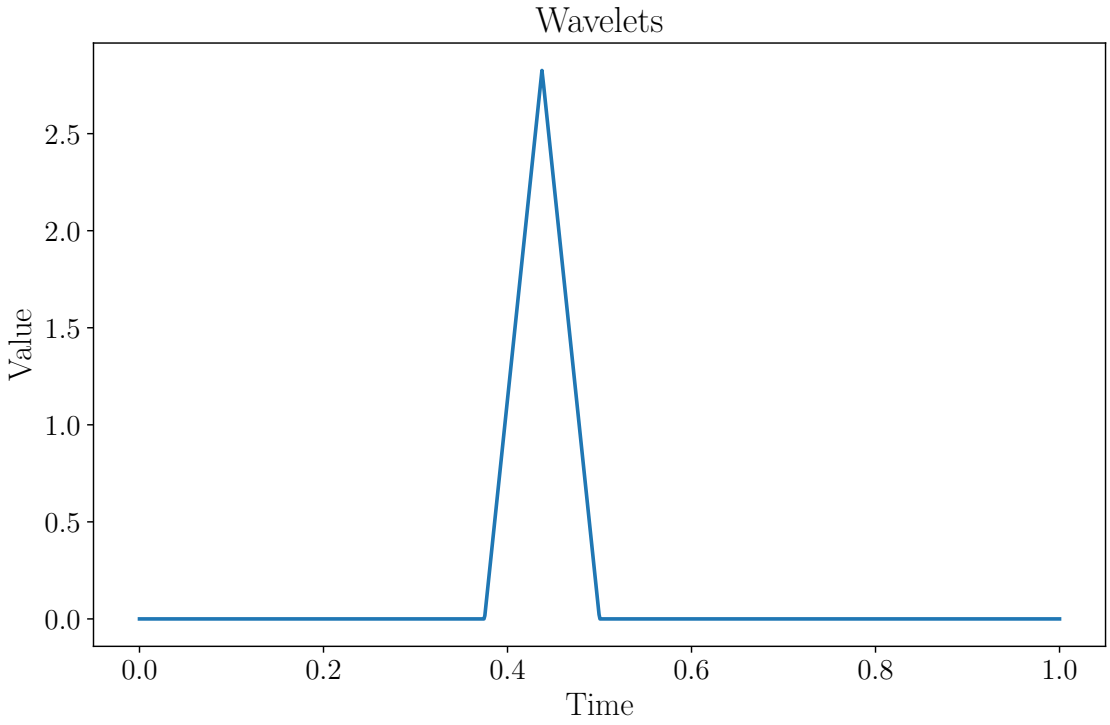

Wavelets
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Value
0.0
0.2
0.4
0.6
0.8
1.0
Time

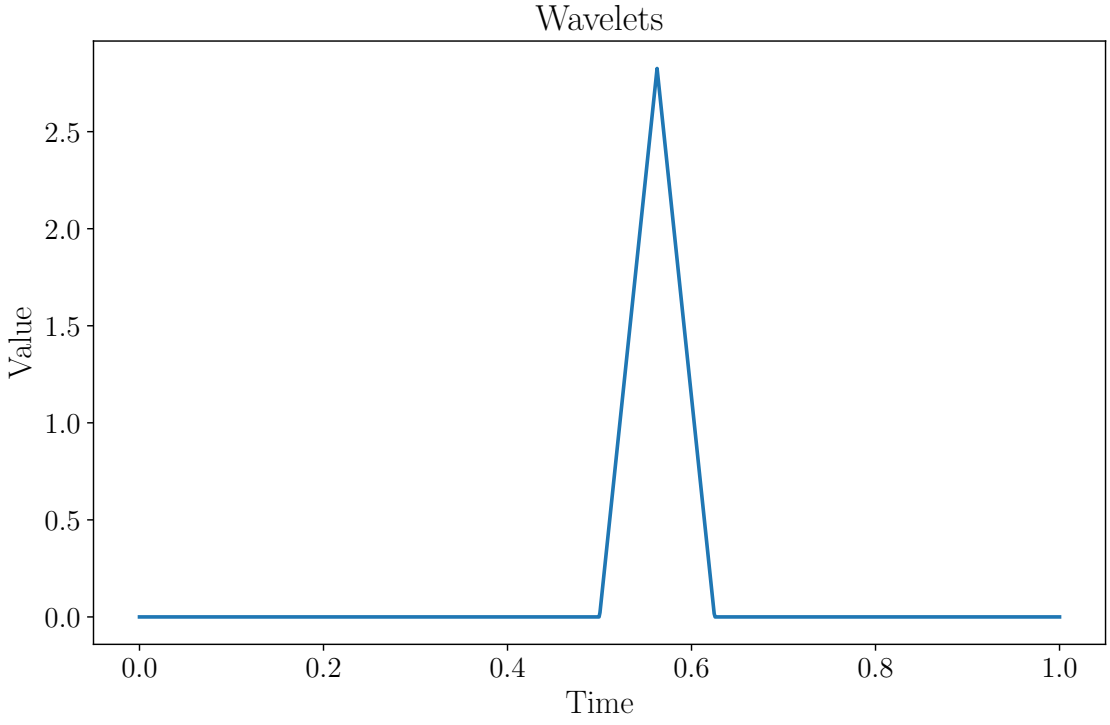

Wavelets
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Value
0.0
0.2
0.4
0.6
0.8
1.0
Time

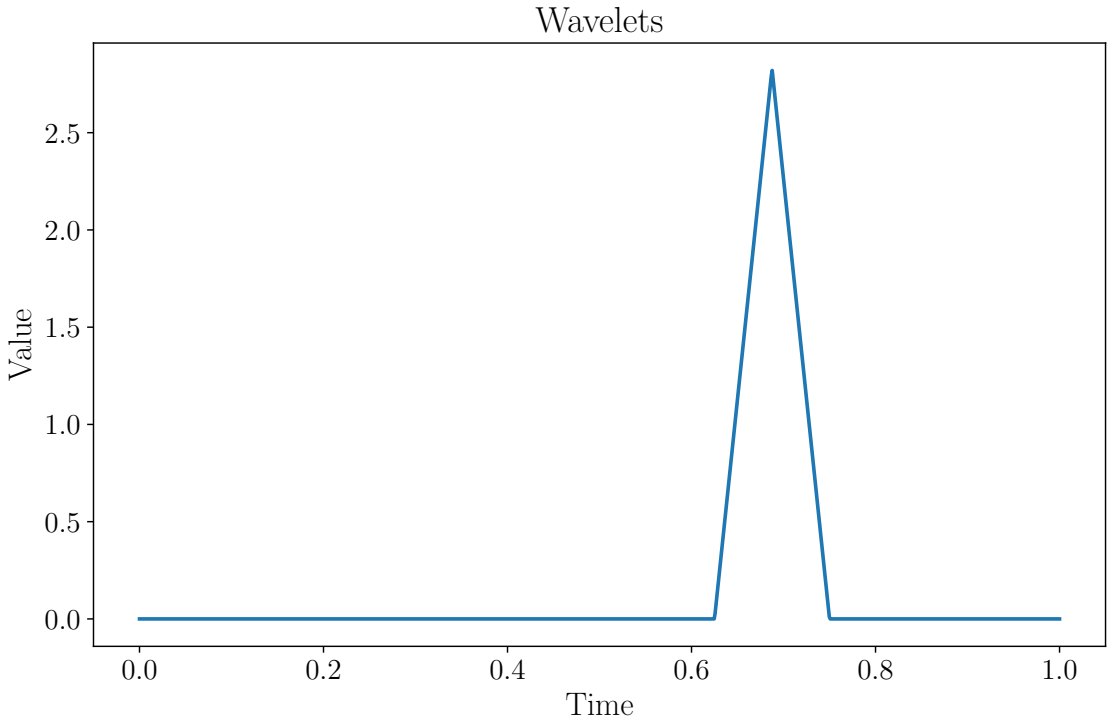

Wavelets
2.5
2.0
1.5
1.0
0.5
0.0
Value
0.0
0.2
0.4
0.6
0.8
1.0
Time

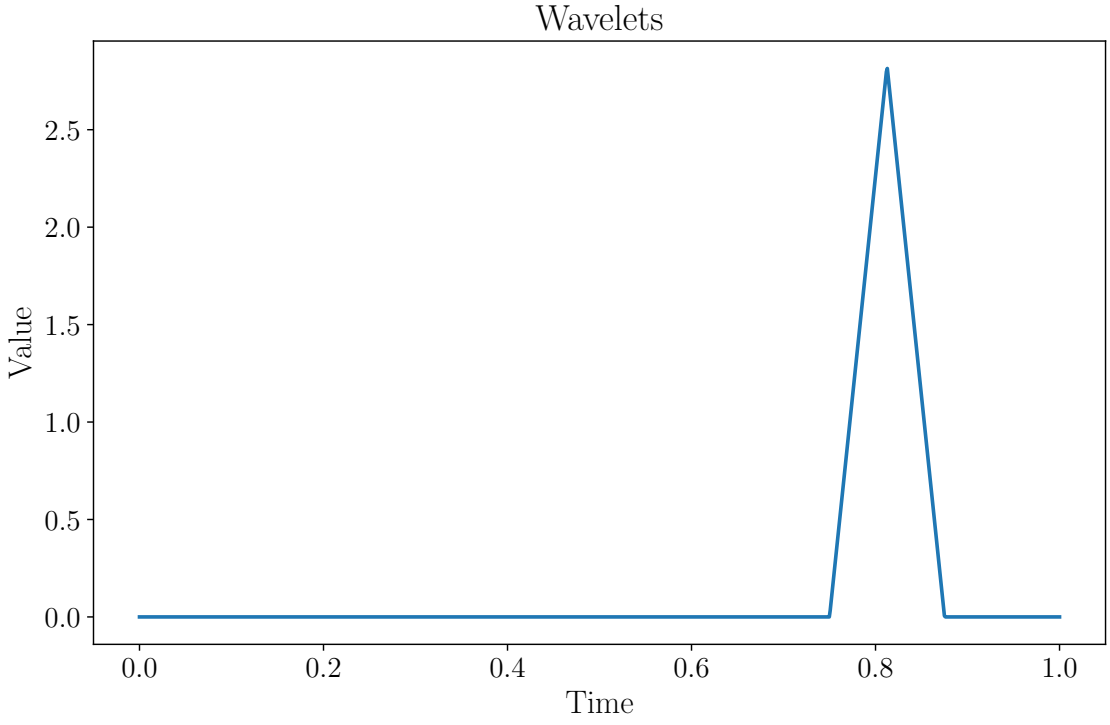

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

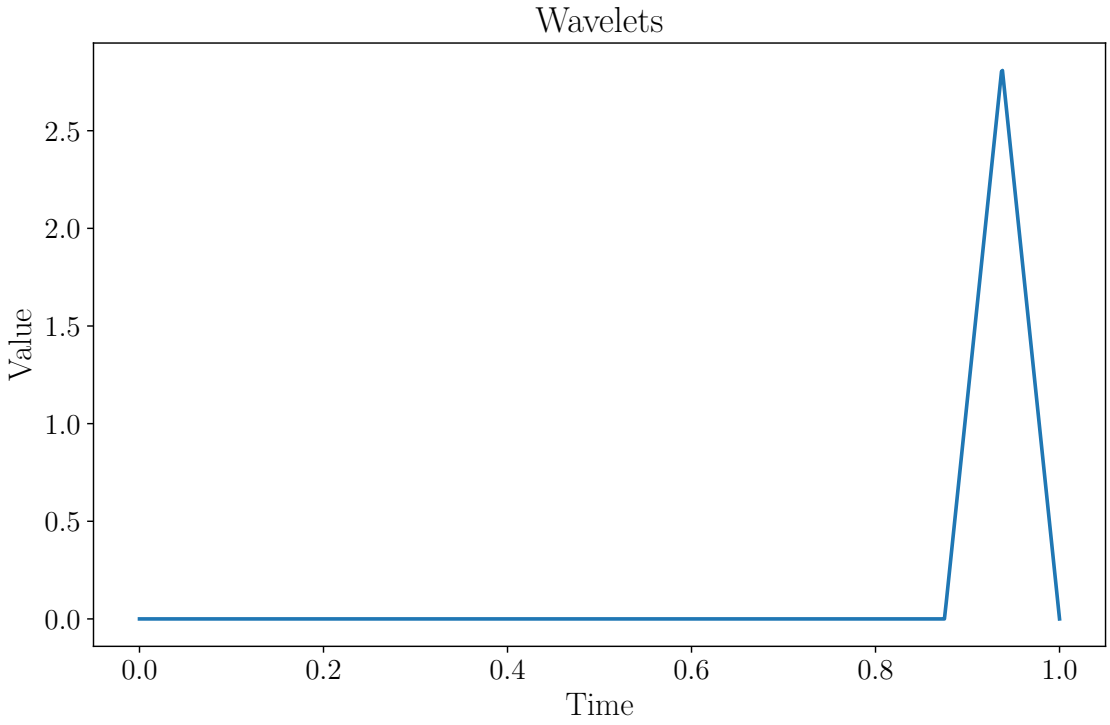

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

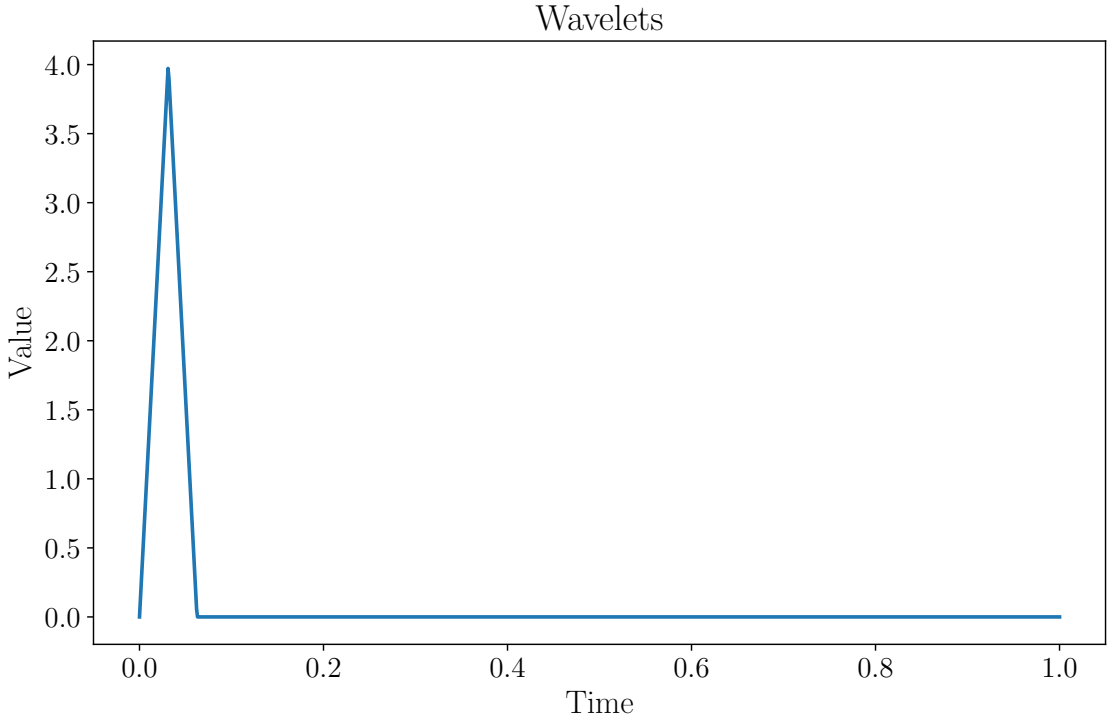

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

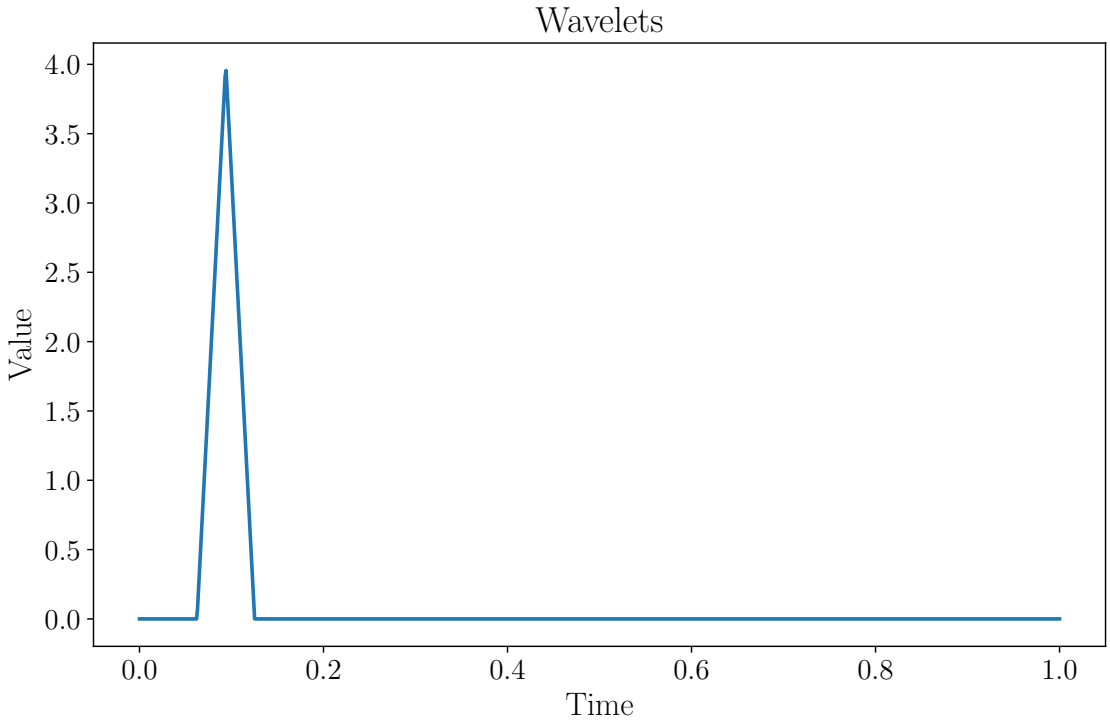

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

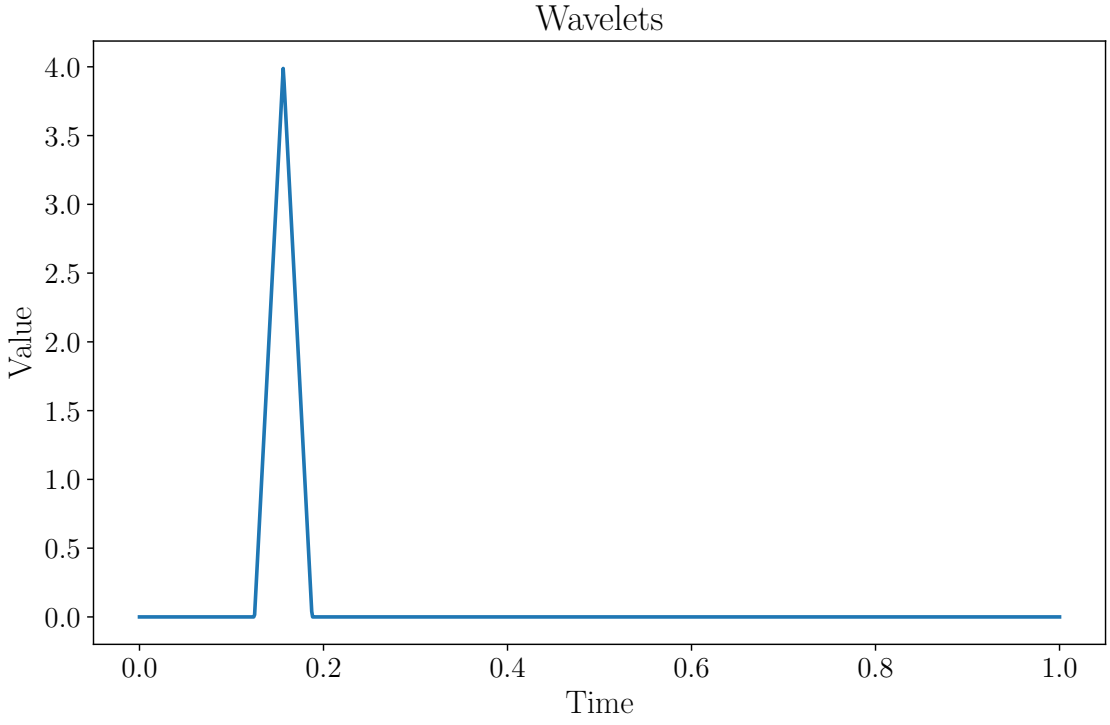

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

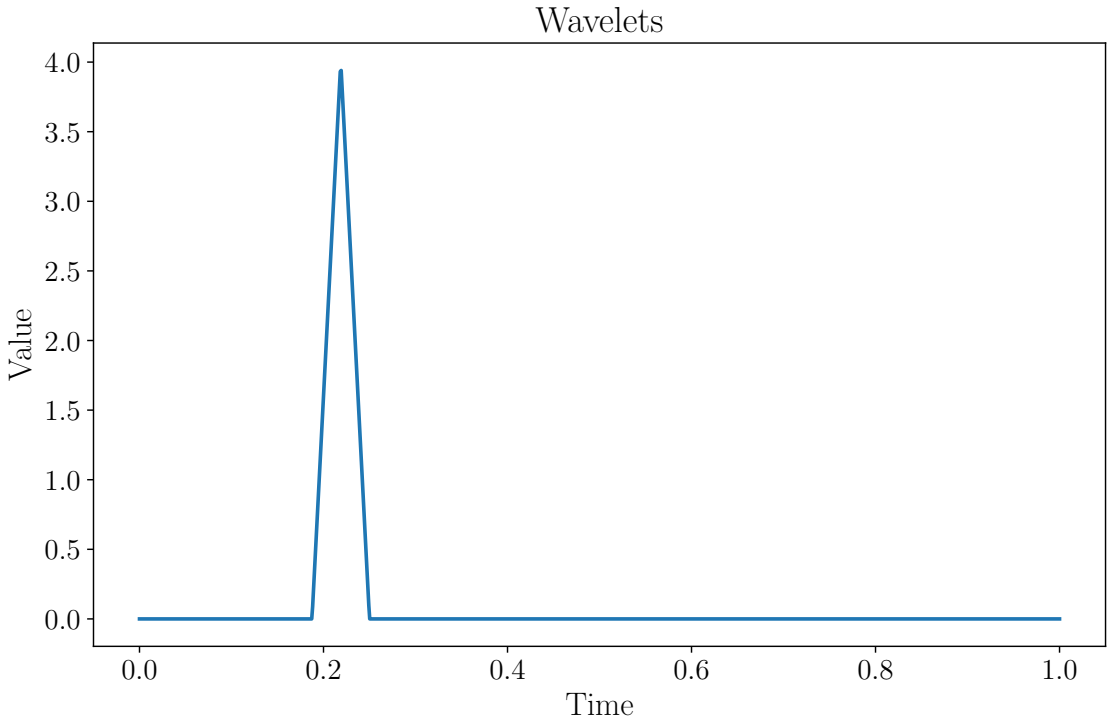

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

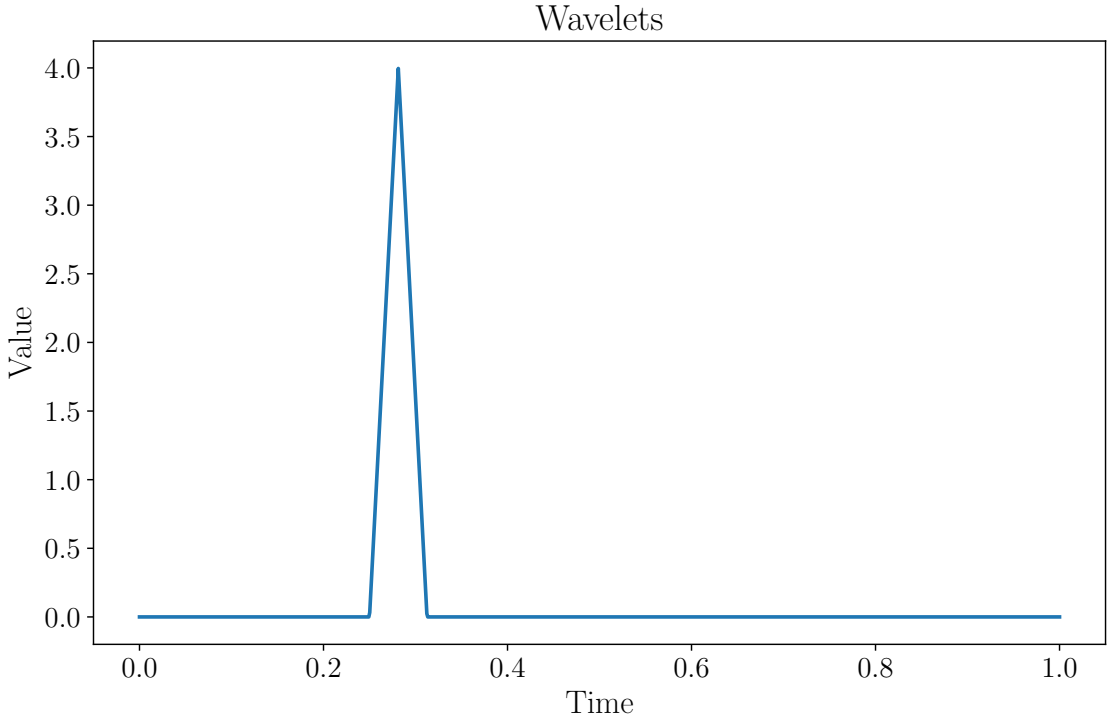

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

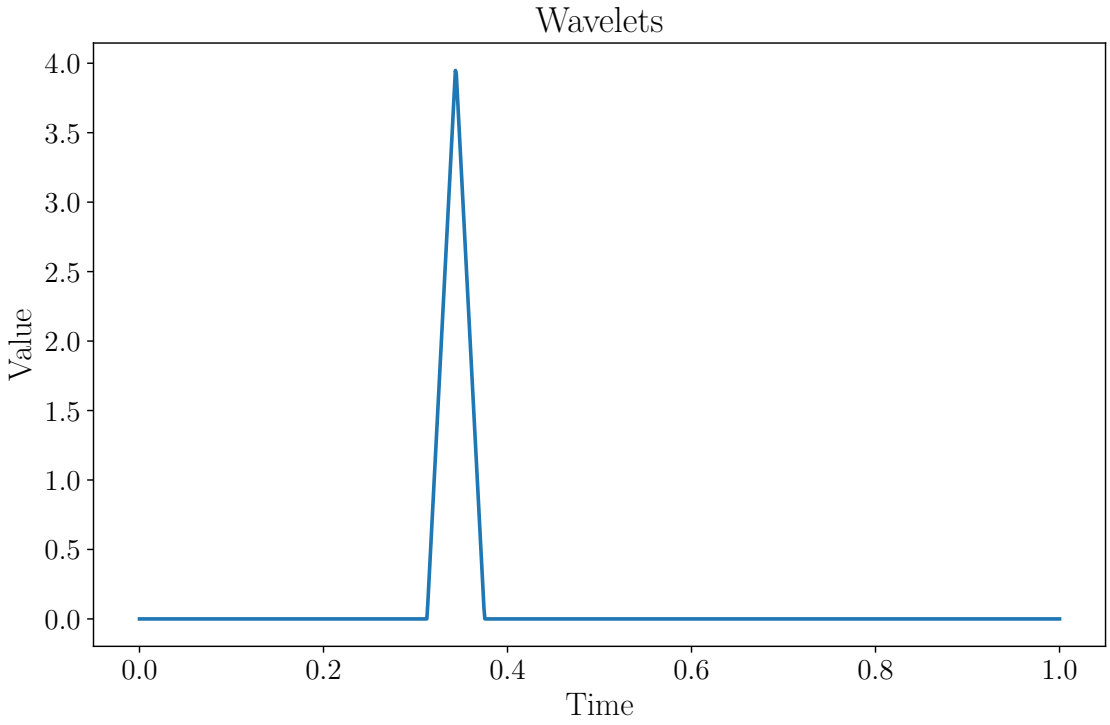

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0
0.2
0.4
0.6
0.8
1.0

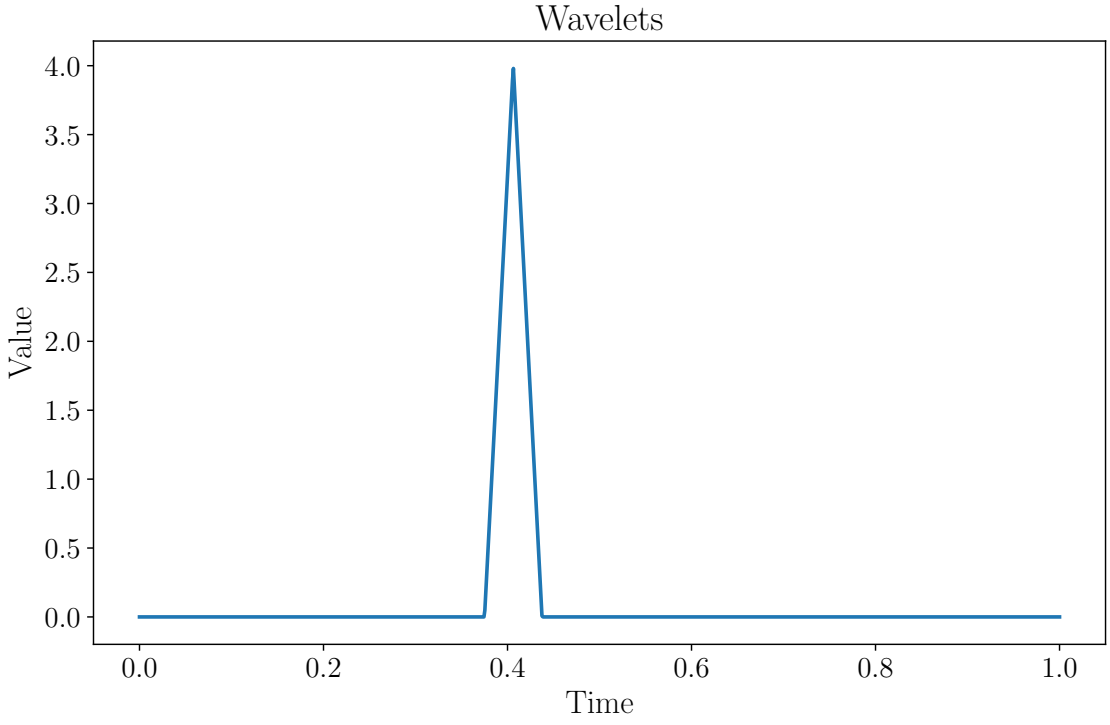

Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

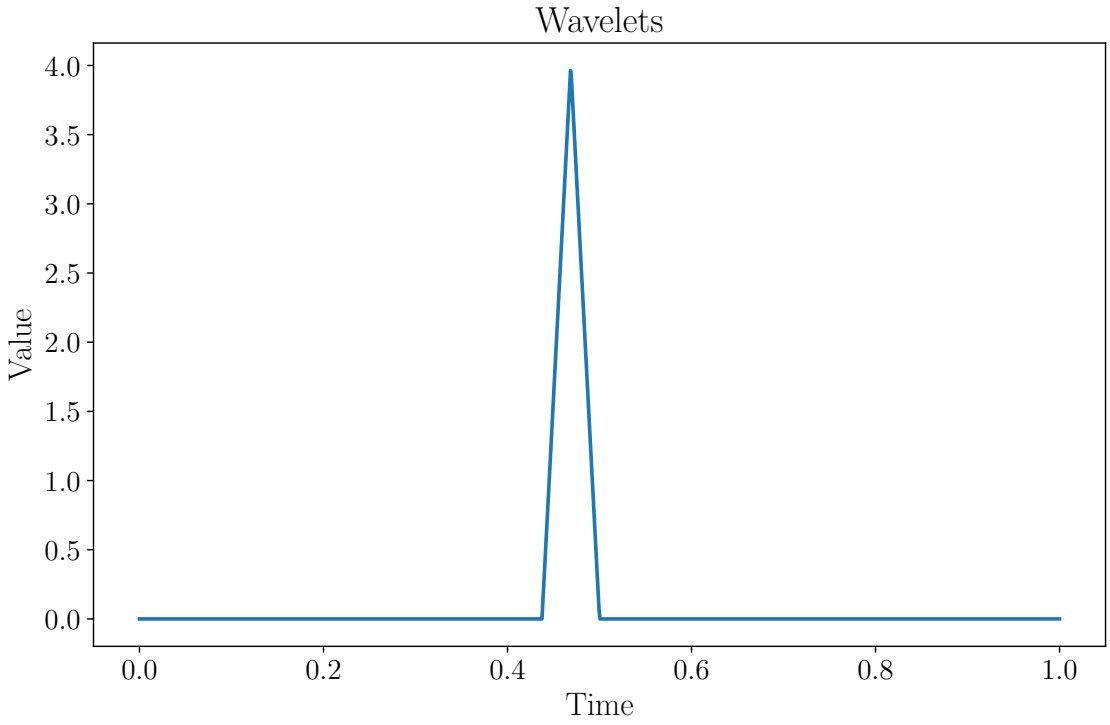

Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

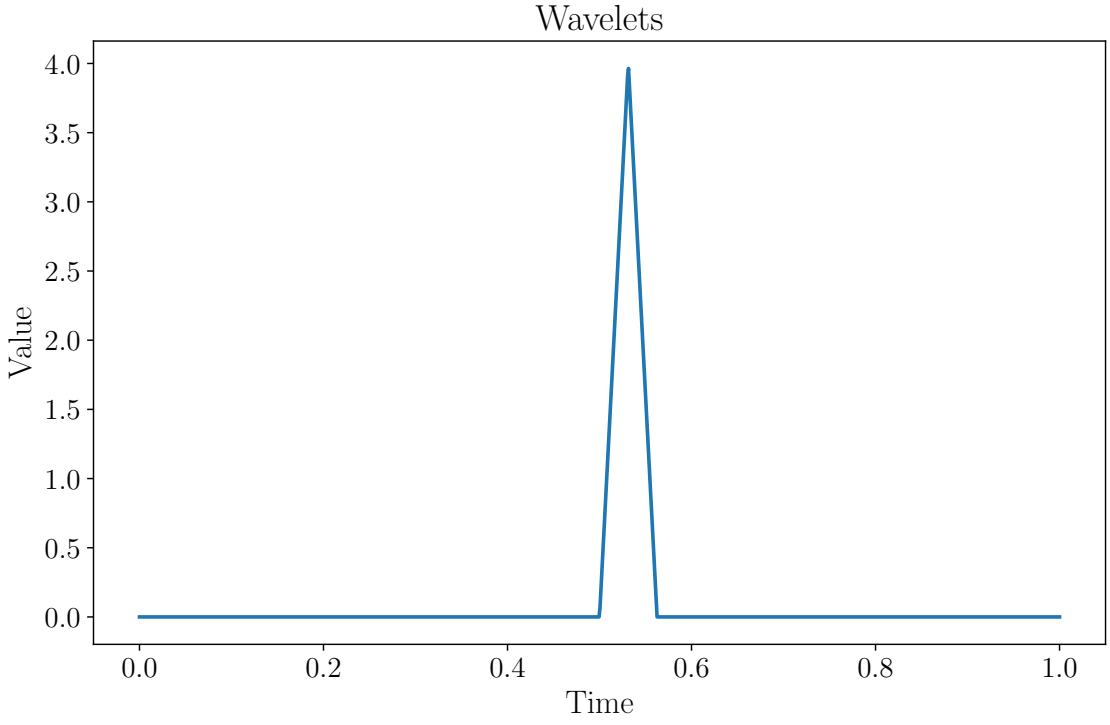

Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

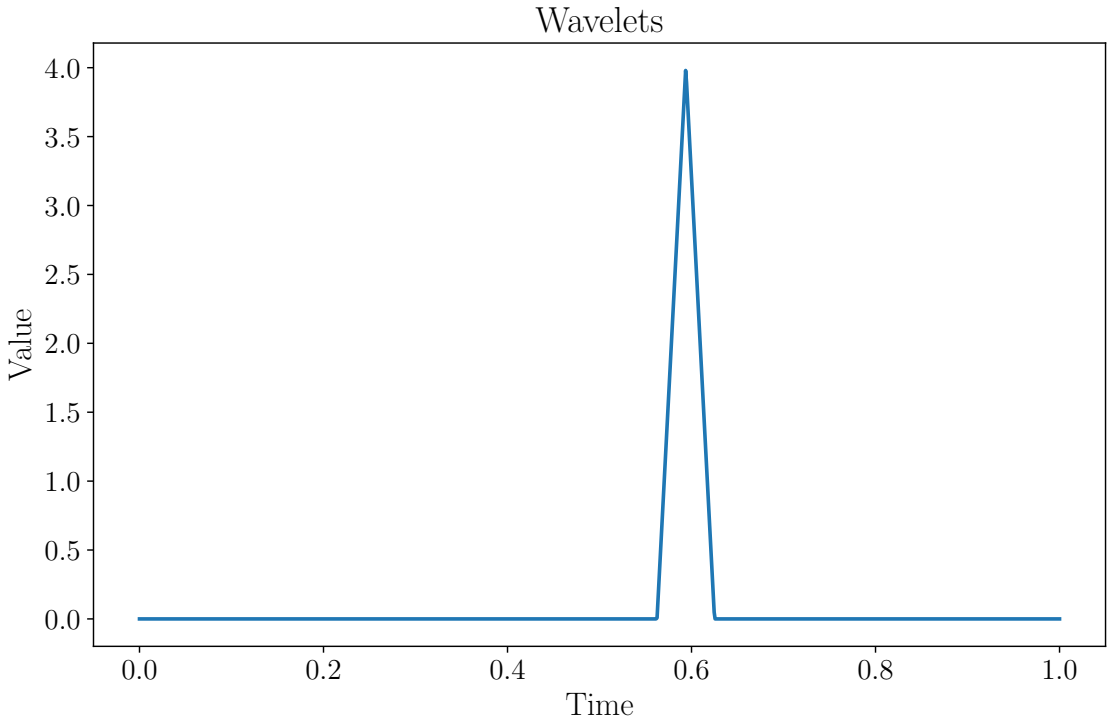

Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

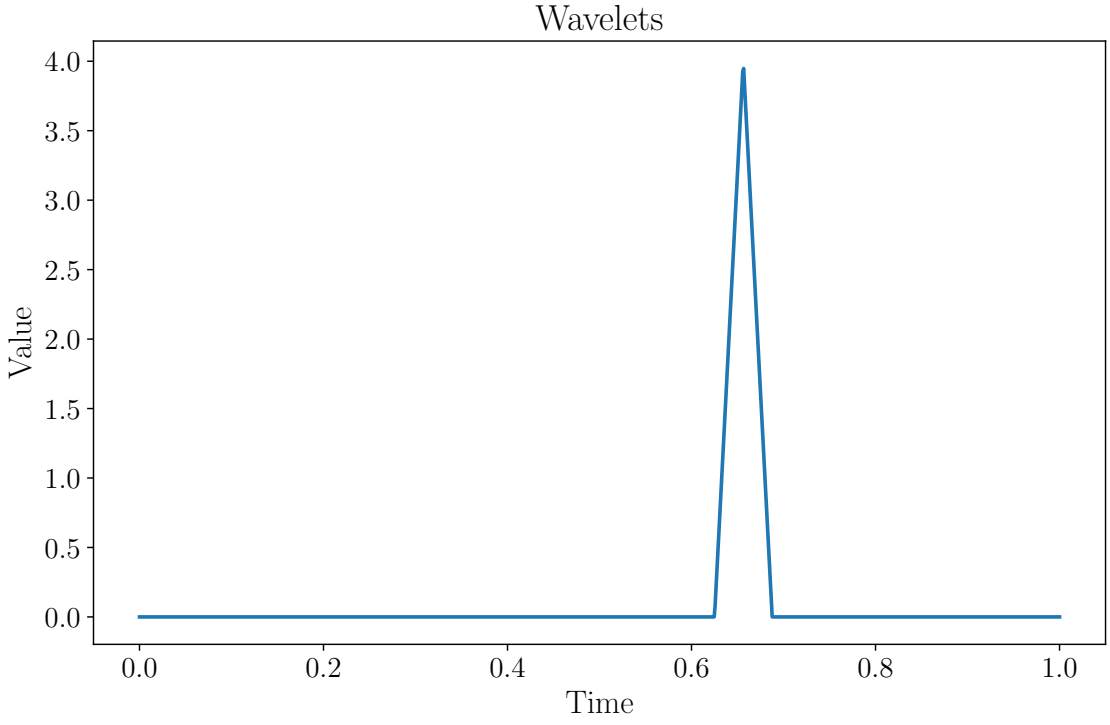

Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

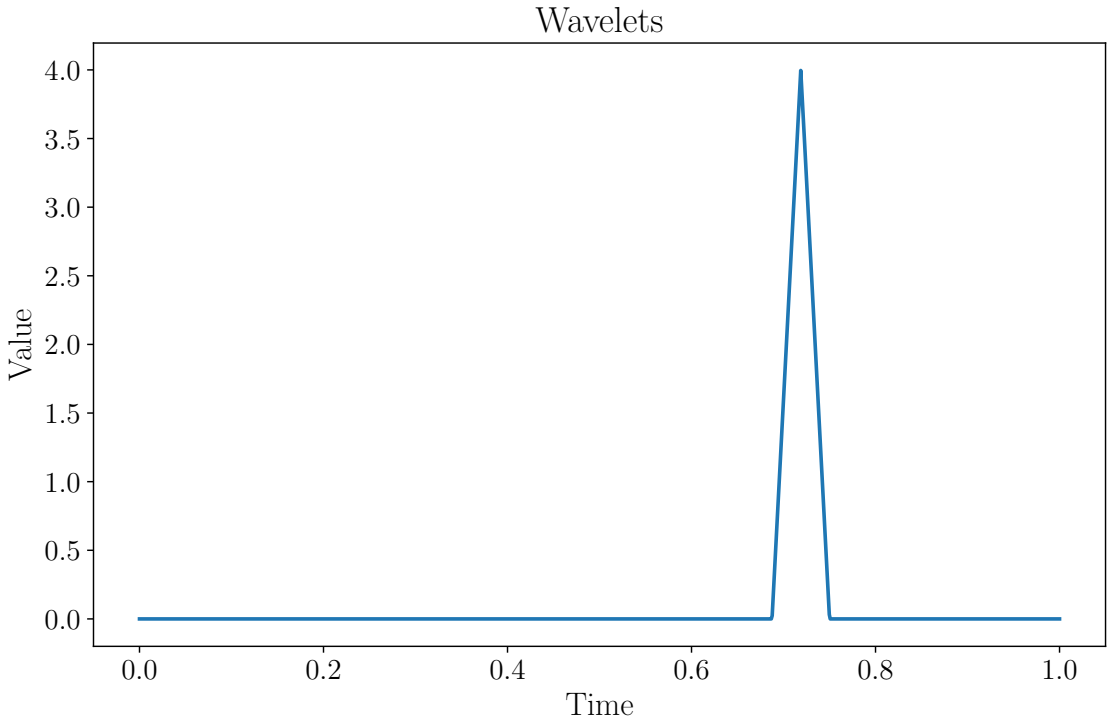

Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

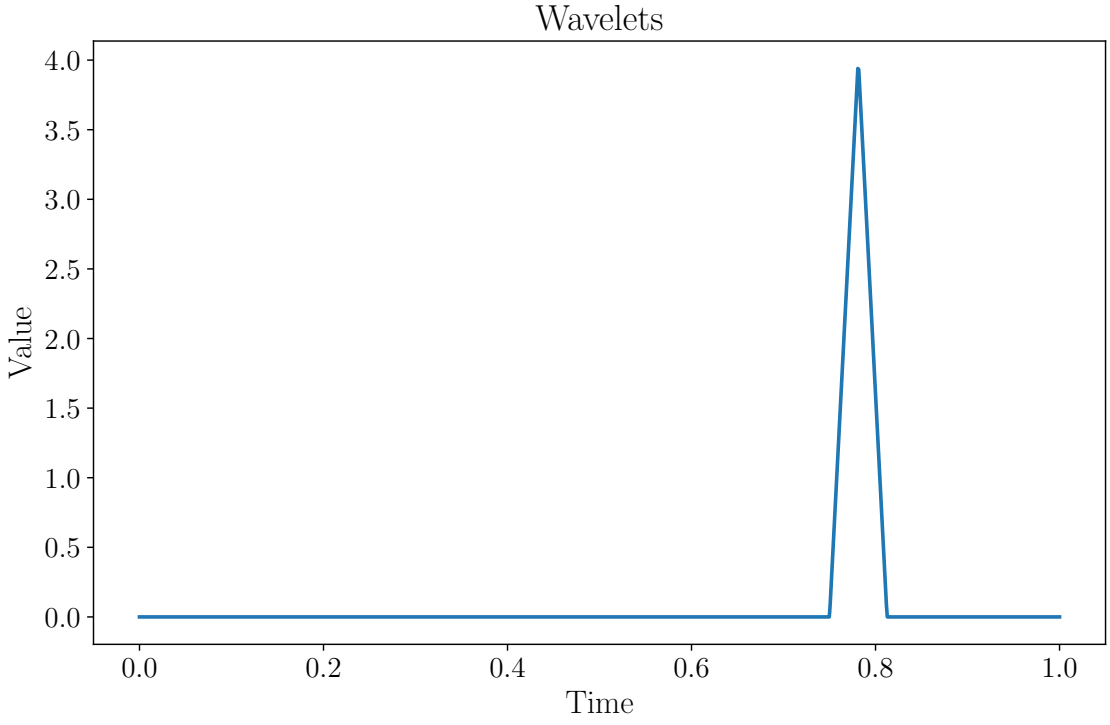

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0
0.2
0.4
0.6
0.8
1.0

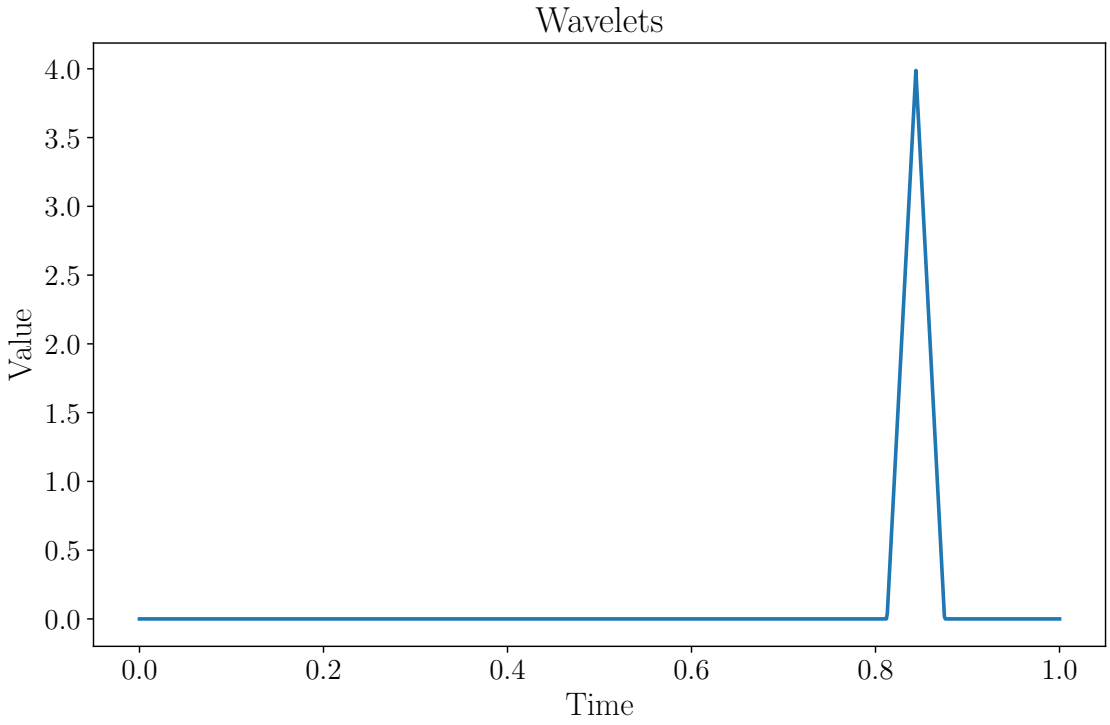

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0
0.2
0.4
0.6
0.8
1.0

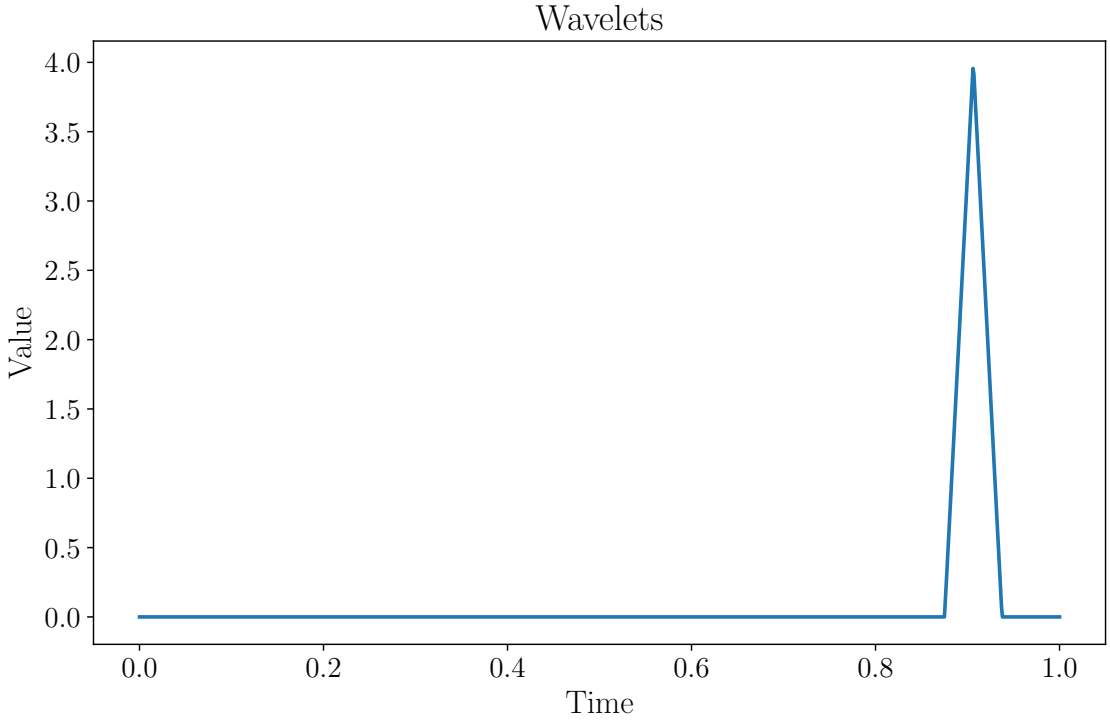

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0
0.2
0.4
0.6
0.8
1.0

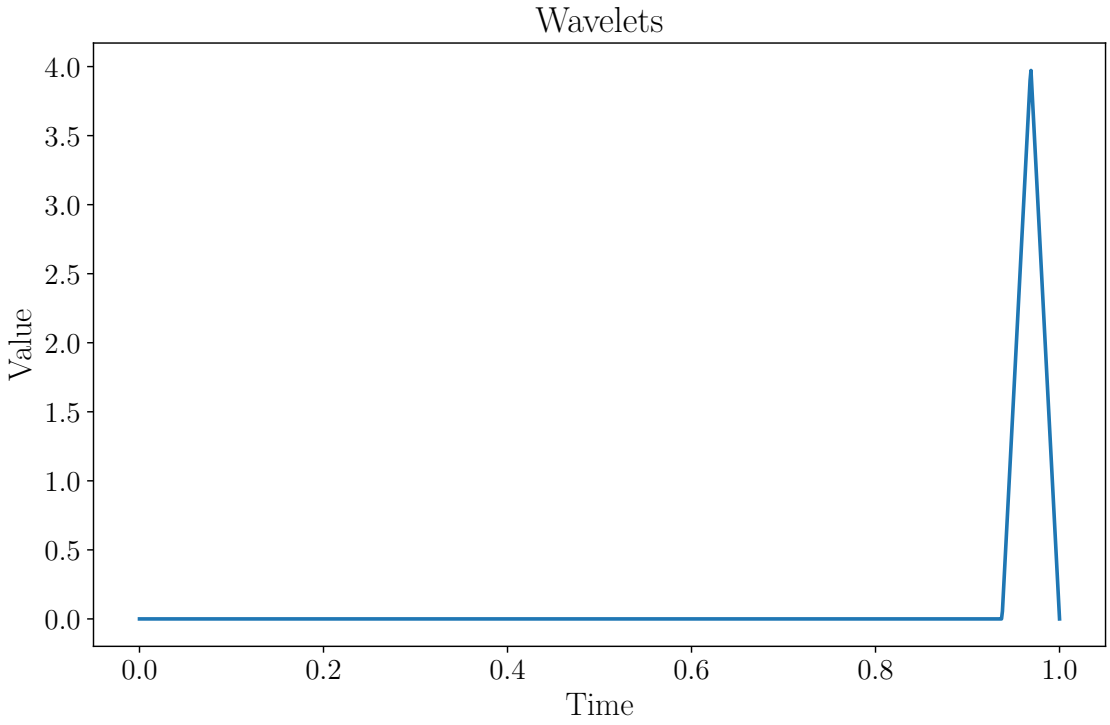

Wavelets
Value
Time
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0
0.2
0.4
0.6
0.8
1.0

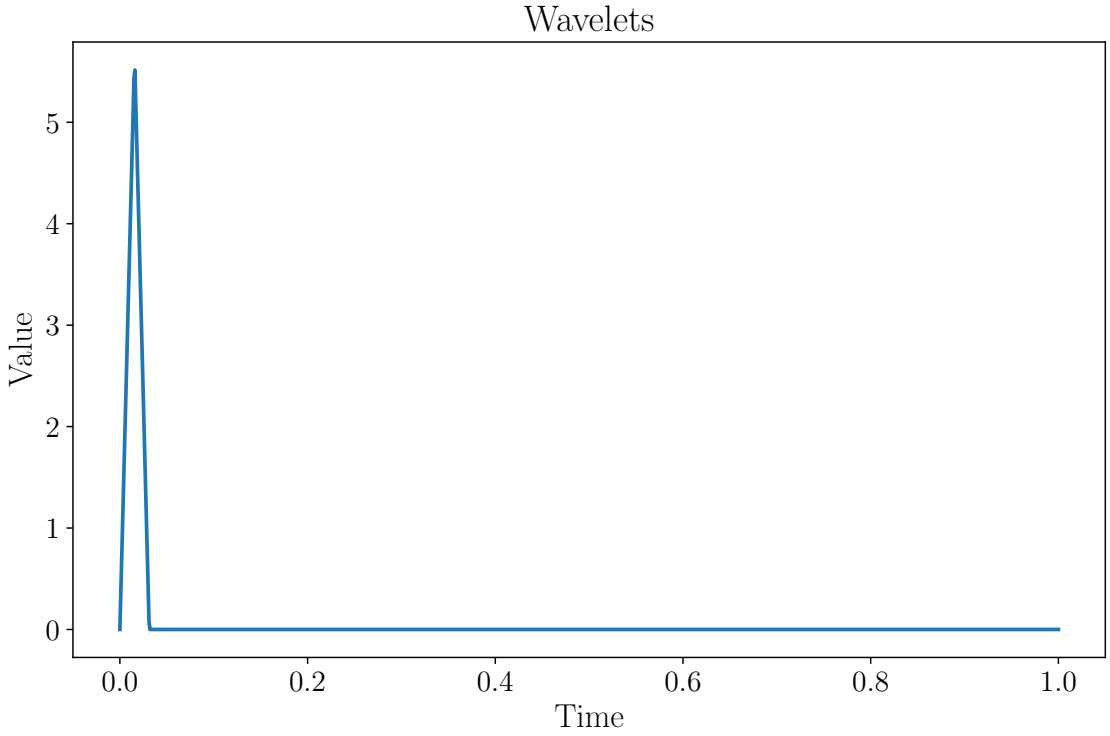

Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Wavelets
Value
Time
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Wavelets
Value
Time
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0