pyfeng

Release 0.1.1

Jaehyuk Choi

Sep 26, 2022

CONTENTS:

PyFENG is the python implemention of the standard option pricing models in financial engineering.

- Black-Scholes-Merton (and displaced diffusion)
- Bachelier (Normal)
- Constant-elasticity-of-variance (CEV)
- Stochastic-alpha-beta-rho (SABR)
- Hyperbolic normal stochastic volatility model (NSVh)

CHAPTER

ONE

ABOUT THE PACKAGE

- It assumes variables are `numpy` arrays. So the computations are naturally vectorized.
- It is purely in Python (i.e., no C, C++, cython).
- It is implemented with python class.
- It is intended for, but not limited to, academic use. By providing reference models, it saves researchers' time.

CHAPTER

TWO

INSTALLATION

```
pip install pyfeng
```

For upgrade,

```
pip install pyfeng --upgrade
```

CHAPTER

THREE

CODE SNIPPETS

In [1]:

```
import numpy as np
import pyfeng as pf
m = pf.Bsm(sigma=0.2, intr=0.05, divr=0.1)
m.price(strike=np.arange(80, 121, 10), spot=100, texp=1.2)
```

Out [1]:

```
array([15.71361973,  9.69250803,  5.52948546,  2.94558338,  1.48139131])
```

In [2]:

```
sigma = np.array([[0.2], [0.5]])
m = pf.Bsm(sigma, intr=0.05, divr=0.1) # sigma in axis=0
m.price(strike=[90, 95, 100], spot=100, texp=1.2, cp=[-1,1,1])
```

Out [2]:

```
array([[ 5.75927238,  7.38869609,  5.52948546],
       [16.812035  , 18.83878533, 17.10541288]])
```

CHAPTER

FOUR

AUTHOR

- Prof. Jaehyuk Choi (Peking University HSBC Business School). Email: pyfe@eml.cc

CHAPTER

FIVE

OTHERS

- See also FER: Financial Engineering in R developed by the same author. Not all models in `PyFENG` is implemented in `FER`. `FER` is a subset of `PyFENG`.

5.1 Black-Scholes-Merton Model

class Bsm(*sigma*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)

Black-Scholes-Merton (BSM) model for option pricing.

Underlying price is assumed to follow a geometric Brownian motion.

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.Bsm(sigma=0.2, intr=0.05, divr=0.1)
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([15.71361973,  9.69250803,  5.52948546,  2.94558338,  1.48139131])
>>> sigma = np.array([0.2, 0.3, 0.5])[:, None]
>>> m = pf.Bsm(sigma, intr=0.05, divr=0.1) # sigma in axis=0
>>> m.price(np.array([90, 100, 110]), 100, 1.2, cp=np.array([-1,1,1]))
array([[ 5.75927238,  5.52948546,  2.94558338],
       [ 9.4592961 ,  9.3881245 ,  6.45745004],
       [16.812035  , 17.10541288, 14.10354768]])
```

cdf(*strike*, *spot*, *texp*, *cp=1*)

Cumulative distribution function of the final asset price.

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – -1 (default) for left-tail CDF, -1 for right-tail CDF (i.e., survival function)

Returns CDF value

d2_var(*strike*, *spot*, *texp*, *cp=1*)

2nd derivative w.r.t. variance (=sigma^2) Eq. (9) in Hull & White (1987)

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns: d^2 price / d var^2

References

- Hull J, White A (1987) The Pricing of Options on Assets with Stochastic Volatilities. The Journal of Finance 42:281–300. https://doi.org/10.1111/j.1540-6261.1987.tb02568.x

d3_var(*strike*, *spot*, *texp*, *cp=1*)
3rd derivative w.r.t. variance (=sigma^2) Eq. (9) in Hull & White (1987)

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns: d^3 price / d var^3

References

- Hull J, White A (1987) The Pricing of Options on Assets with Stochastic Volatilities. The Journal of Finance 42:281–300. https://doi.org/10.1111/j.1540-6261.1987.tb02568.x

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns gamma value

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
BSM implied volatility with Newton's method

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

moments_vsk(*texp=1*)
Variance, skewness, and ex-kurtosis. Assume mean=1.

Parameters **texp** – time-to-expiry

Returns (variance, skewness, and ex-kurtosis)

References

https://en.wikipedia.org/wiki/Log-normal_distribution

price_barrier(*strike*, *barrier*, *spot*, *texp*, *cp=1*, *io=- 1*)
Barrier option price under the BSM model

Parameters

- **strike** – strike price
- **barrier** – knock-in/out barrier price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **io** – +1 for knock-in, -1 for knock-out

Returns Barrier option price

static price_formula(*strike*, *spot*, *sigma*, *texp*, *cp=1*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Black-Scholes-Merton model call/put option pricing formula (static method)

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **sigma** – model volatility
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **intr** – interest rate (domestic interest rate)
- **divr** – dividend/convenience yield (foreign interest rate)

- **is_fwd** – if True, treat *spot* as forward price. False by default.

Returns Vanilla option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model theta (sensitivity to time-to-maturity).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns theta value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model='norm'*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'norm' (default), 'norm-approx', 'norm-grunspan', 'bsm'}

Returns volatility smile under the specified model

class BsmDisp(*sigma*, *beta=1*, *pivot=0*, **args*, ***kwargs*)
Displaced Black-Scholes-Merton model for option pricing. Displace price,

D(F_t) = beta*F_t + (1-beta)*A

is assumed to follow a geometric Brownian motion with volatility *beta*sigma*.

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.BsmDisp(sigma=0.2, beta=0.5, pivot=100, intr=0.05, divr=0.1)
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([15.9543935 ,  9.7886658 ,  5.4274197 ,  2.71430505,  1.22740381])
>>> beta = np.array([0.2, 0.6, 1])[:, None]
>>> m = pf.BsmDisp(0.2, beta=beta, pivot=100) # beta in axis=0
>>> m.vol_smile(np.arange(80, 121, 10), 100, 1.2)
array([[0.21915778, 0.20904587, 0.20038559, 0.19286293, 0.18625174],
       [0.20977955, 0.20461468, 0.20025691, 0.19652101, 0.19327567],
       [0.2       , 0.2       , 0.2       , 0.2       , 0.2       ]])
```

cdf (*strike*, *spot*, *texp*, *cp=1*)
Cumulative distribution function of the final asset price.

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – -1 (default) for left-tail CDF, -1 for right-tail CDF (i.e., survival function)

Returns CDF value

delta (*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

disp_spot (*spot*)
Displaced spot

Parameters **spot** – spot (or forward) price

Returns Displaces spot

disp_strike (*strike*, *texp*)
Displaced strike

Parameters

- **strike** – strike price
- **texp** – time to expiry

Returns Displace strike

gamma (*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price_in*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

moments_vsk(*texp=1*)
Variance, skewness, and ex-kurtosis. Assume mean=1.

Parameters **texp** – time-to-expiry

Returns (variance, skewness, and ex-kurtosis)

References

https://en.wikipedia.org/wiki/Log-normal_distribution

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

price_barrier(*strike*, *barrier*, *spot*, **args*, ***kwargs*)
Barrier option price under the BSM model

Parameters

- **strike** – strike price
- **barrier** – knock-in/out barrier price
- **spot** – spot price
- **texp** – time to expiry

- **cp** – 1/-1 for call/put option
- **io** – +1 for knock-in, -1 for knock-out

Returns Barrier option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model='bsm'*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm', 'bsm-approx', 'norm-approx'}

Returns volatility smile under the specified model

5.2 Constant Elasticity of Variance (CEV) Model

class Cev(*sigma*, *beta=0.5*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Constant Elasticity of Variance (CEV) model.

Underlying price is assumed to follow CEV process: dS_t = (r - q) S_t dt + sigma S_t^beta dW_t, where dW_t is a standard Brownian motion.

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.Cev(sigma=0.2, beta=0.5, intr=0.05, divr=0.1)
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([16.11757214, 10.00786871,  5.64880408,  2.89028476,  1.34128656])
```

cdf (*strike*, *spot*, *texp*, *cp=1*)
Cumulative distribution function of the final asset price.

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – -1 (default) for left-tail CDF, -1 for right-tail CDF (i.e., survival function)

Returns CDF value

delta (*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

gamma (*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns gamma value

mass_zero (*spot*, *texp*, *log=False*)
Probability mass absorbed at the zero boundary (K=0)

Parameters

- **spot** – spot (or forward) price
- **texp** – time to expiry
- **log** – log value if True

Returns (log) probability mass at zero

mass_zero_t0(*spot*, *texp*)
Limit value of -T log(M_T) as T -> 0, where M_T is the mass at zero.

Parameters **spot** – spot (or forward) price

Returns

- lim_{T->0} T log(M_T)

params_kw()
Model parameters in dictionary

static price_formula(*strike*, *spot*, *texp*, *sigma=None*, *cp=1*, *beta=0.5*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
CEV model call/put option pricing formula (static method)

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **sigma** – model volatility
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **beta** – elasticity parameter
- **intr** – interest rate (domestic interest rate)
- **divr** – dividend/convenience yield (foreign interest rate)
- **is_fwd** – if True, treat *spot* as forward price. False by default.

Returns Vanilla option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model theta (sensitivity to time-to-maturity).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns theta value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

5.3 Bachelier (Normal) Model

Created on Tue Sep 19 22:56:58 2017 @author: jaehyuk

class Norm(*sigma*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)

Bachelier (normal) model for option pricing. Underlying price is assumed to follow arithmetic Brownian motion.

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.Norm(sigma=20, intr=0.05, divr=0.1)
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([16.57233446, 10.34711401,  5.77827026,  2.83857367,  1.20910477])
>>> sigma = np.array([20, 30, 50])[:, None]
>>> m = pf.Norm(sigma, intr=0.05, divr=0.1) # sigma in axis=0
>>> m.price(np.array([90, 100, 110]), 100, 1.2, cp=np.array([-1,1,1]))
array([[ 6.41387836,  5.77827026,  2.83857367],
       [10.48003559,  9.79822867,  6.3002881 ],
       [18.67164469, 17.95246828, 13.98027179]])
```

cdf(*strike*, *spot*, *texp*, *cp=1*)

Cumulative distribution function of the final asset price.

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – -1 (default) for left-tail CDF, -1 for right-tail CDF (i.e., survival function)

Returns CDF value

delta(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

gamma(*strike*, *spot*, *texp*, *cp=1*)

Option model gamma (2nd derivative to price).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry

- **cp** – 1/-1 for call/put option

Returns gamma value

impvol (*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Bachelier implied volatility by Choi et al. (2007)

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

References

Choi, J., Kim, K., & Kwak, M. (2009). Numerical Approximation of the Implied Volatility Under Arithmetic Brownian Motion. Applied Mathematical Finance, 16(3), 261–268. https://doi.org/10.1080/13504860802583436

Returns implied volatility

price_barrier (*strike*, *barrier*, *spot*, *texp*, *cp=1*, *io=- 1*)
Barrier option price under the BSM model

Parameters

- **strike** – strike price
- **barrier** – knock-in/out barrier price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **io** – +1 for knock-in, -1 for knock-out

Returns Barrier option price

static price_formula (*strike*, *spot*, *sigma*, *texp*, *cp=1*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Bachelier model call/put option pricing formula (static method)

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **sigma** – model volatility
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **sigma** – model volatility
- **intr** – interest rate (domestic interest rate)
- **divr** – dividend/convenience yield (foreign interest rate)

- **is_fwd** – if True, treat *spot* as forward price. False by default.

Returns Vanilla option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model theta (sensitivity to time-to-maturity).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns theta value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility).

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model='bsm'*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm' (default), 'bsm', 'bsm-approx', 'norm'}

Returns volatility smile under the specified model

5.4 Hyperbolic Normal Stochastic Volatility (NSVh) Model

class Nsvh1(*sigma*, *vov=0.0*, *rho=0.0*, *beta=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*, *is_atmvol=False*)
Hyperbolic Normal Stochastic Volatility (NSVh) model with lambda=1 by Choi et al. (2019)

References

- Choi J, Liu C, Seo BK (2019) Hyperbolic normal stochastic volatility model. J Futures Mark 39:186–204. https://doi.org/10.1002/fut.21967

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.Nsvh1(sigma=20, vov=0.8, rho=-0.3)
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([25.51200027, 17.87539874, 11.47308947,  6.75128331,  3.79464422])
```

calibrate_vsk(*var*, *skew*, *exkurt*, *texp=1*, *setval=False*)

Calibrate parameters to the moments: variance, skewness, ex-kurtosis.

Parameters

- **texp** – time-to-expiry
- **var** – variance
- **skew** – skewness
- **exkurt** – ex-kurtosis. should be > 0.

Returns: (sigma, vov, rho)

References

Tuenter, H. J. H. (2001). An algorithm to determine the parameters of SU-curves in the johnson system of probabillity distributions by moment matching. Journal of Statistical Computation and Simulation, 70(4), 325–347. https://doi.org/10.1080/00949650108812126

moments_vsk(*texp=1*)

Variance, skewness, and ex-kurtosis

Parameters **texp** – time-to-expiry

Returns (variance, skewness, and ex-kurtosis)

price(*strike*, *spot*, *texp*, *cp=1*)

Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

class NsvhMc(*sigma*, *vov=0.0*, *rho=0.0*, *lam=0.0*, *beta=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)

Monte-Carlo model of Hyperbolic Normal Stochastic Volatility (NSVh) model.

NSVh with lambda = 0 is the normal SABR model, and NSVh with lambda = 1 has analytic pricing (Nsvh1)

References

- Choi J, Liu C, Seo BK (2019) Hyperbolic normal stochastic volatility model. J Futures Mark 39:186–204. https://doi.org/10.1002/fut.21967

See also:

Nsvh1

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.NsvhMc(sigma=20, vov=0.8, rho=-0.3, lam=0.0)
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([23.52722081, 15.63212633,  9.19644639,  4.81061848,  2.39085097])
>>> m1 = pf.NsvhMc(sigma=20, vov=0.8, rho=-0.3, lam=1.0)
>>> m2 = pf.Nsvh1(sigma=20, vov=0.8, rho=-0.3)
>>> p1 = m1.price(np.arange(80, 121, 10), 100, 1.2)
>>> p2 = m2.price(np.arange(80, 121, 10), 100, 1.2)
>>> p1 - p2
array([-0.00328887,  0.00523714,  0.00808885,  0.0069694 ,  0.00205566])
```

mc_vol_price(*texp*)
Simulate volatility and price pair

Parameters **texp** – time-to-expiry

Returns: (vol, price). vol: (n_path,), price: (n_path, 2)

price(*strike*, *spot*, *texp*, *cp=1*)
Vanilla option price from MC simulation of NSVh model.

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to np.expiry
- **cp** – 1/-1 for call/put

Returns vanilla option price

class NsvhQuadInt(*sigma*, *vov=0.0*, *rho=0.0*, *lam=0.0*, *beta=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Quadrature integration method of Hyperbolic Normal Stochastic Volatility (NSVh) model.

NSVh with lambda = 0 is the normal SABR model, and NSVh with lambda = 1 has analytic pricing (Nsvh1)

References

- Choi J, Liu C, Seo BK (2019) Hyperbolic normal stochastic volatility model. J Futures Mark 39:186–204. https://doi.org/10.1002/fut.21967

See also:

Nsvh1, SabrNormalVolApprox

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> #### Nsvh1: comparison with analytic pricing
>>> m1 = pf.NsvhQuadInt(sigma=20, vov=0.8, rho=-0.3, lam=1.0)
>>> m2 = pf.Nsvh1(sigma=20, vov=0.8, rho=-0.3)
>>> p1 = m1.price(np.arange(80, 121, 10), 100, 1.2)
>>> p2 = m2.price(np.arange(80, 121, 10), 100, 1.2)
>>> p1 - p2
array([0.00345526, 0.00630649, 0.00966333, 0.00571175, 0.00017924])
>>> #### Normal SABR: comparison with vol approximation
>>> m1 = pf.NsvhQuadInt(sigma=20, vov=0.8, rho=-0.3, lam=0.0)
>>> m2 = pf.SabrNormVolApprox(sigma=20, vov=0.8, rho=-0.3)
>>> p1 = m1.price(np.arange(80, 121, 10), 100, 1.2)
>>> p2 = m2.price(np.arange(80, 121, 10), 100, 1.2)
>>> p1 - p2
array([-0.17262802, -0.10160687, -0.00802731,  0.0338126 ,  0.01598512])
```

References

Choi J (2023), Unpublished Working Paper.

price(*strike*, *spot*, *texp*, *cp=1*)

Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

5.5 Stochastic-Alpha-Beta-Rho (SABR) Model

Created on Tue Oct 10 @author: jaehyuk

class SabrChoiWu2021H(*sigma*, *vov=0.0*, *rho=0.0*, *beta=1.0*, *intr=0.0*, *divr=0.0*, *is_fwd=False*, *vol_beta=None*)

The CEV volatility approximation of the SABR model based on Theorem 1 of Choi & Wu (2019)

References

- Choi, J., & Wu, L. (2019). The equivalent constant-elasticity-of-variance (CEV) volatility of the stochastic-alpha-beta-rho (SABR) model. ArXiv:1911.13123 [q-Fin]. https://arxiv.org/abs/1911.13123

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.SabrChoiWu2021H(sigma=2, vov=0.2, rho=-0.3, beta=0.5)
>>> m.vol_for_price(np.arange(80, 121, 10), 100, 1.2)
array([2.07833214, 2.03698255, 2.00332   , 1.97692259, 1.95735019])
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([22.04897988, 14.56240351,  8.74169054,  4.72340753,  2.28876105])
```

base_model(*vol*, *is_fwd=None*)

Create base model based on _base_beta value: *Norm* for 0, Cev for (0,1), and *Bsm* for 1 If *_base_beta* is None, use *base* instead.

Parameters

- **vol** – base model volatility
- **is_fwd** – if True, treat *spot* as forward price. False by default.

Returns: model

delta(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry

- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_benchmark(*set_no=None*)

Initiate a SABR model with stored benchmark parameter sets

Parameters **set_no** – set number

Returns Dataframe of all test cases if set_no = None (model, Dataframe of result, params) if set_no is specified

References

- Antonov, Alexander, Konikov, M., & Spector, M. (2013). SABR spreads its wings. Risk, 2013(Aug), 58–63.
- Antonov, Alexandre, Konikov, M., & Spector, M. (2019). Modern SABR Analytics. Springer International Publishing. https://doi.org/10.1007/978-3-030-10656-0
- Antonov, Alexandre, & Spector, M. (2012). Advanced analytics for the SABR model. Available at SSRN. https://ssrn.com/abstract=2026350
- Cai, N., Song, Y., & Chen, N. (2017). Exact Simulation of the SABR Model. Operations Research, 65(4), 931–951. https://doi.org/10.1287/opre.2017.1617
- Korn, R., & Tang, S. (2013). Exact analytical solution for the normal SABR model. Wilmott Magazine, 2013(7), 64–69. https://doi.org/10.1002/wilm.10235
- Lewis, A. L. (2016). Option valuation under stochastic volatility II: With Mathematica code. Finance Press.
- von Sydow, L., …, Haentjens, T., & Waldén, J. (2018). BENCHOP - SLV: The BENCHmarking project in Option Pricing – Stochastic and Local Volatility problems. International Journal of Computer Mathematics, 1–14. https://doi.org/10.1080/00207160.2018.1544368

mass_zero(*spot*, *texp*, *log=False*)

Probability mass absorbed at the zero boundary (K=0)

Parameters

- **spot** – spot (or forward) price
- **texp** – time to expiry
- **log** – log value if True

Returns (log) probability mass at zero

mass_zero_t0(*spot*, *texp*)

Limit value of -T log(M_T) as T -> 0, where M_T is the mass at zero. See Corollary 3.1 of Choi & Wu (2019)

Parameters

- **spot** – spot (or forward) price

- **texp** – time to expiry

Returns -lim_{T->0} T log(M_T)

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_for_price(*strike*, *spot*, *texp*)
Equivalent volatility of the SABR model

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **texp** – time to expiry

Returns equivalent volatility

vol_from_mass_zero(*strike*, *spot*, *texp*, *mass=None*)
Implied volatility from positive mass at zero from DMHJ (2017) If mass is given, use the given value. If None (by default), compute model implied value.

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **mass** – probability mass at zero (None by default)

Returns implied BSM volatility

References

De Marco, S., Hillairet, C., & Jacquier, A. (2017). Shapes of Implied Volatility with Positive Mass at Zero. SIAM Journal on Financial Mathematics, 8(1), 709–737. https://doi.org/10.1137/14098065X

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model=None*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm'} 'bsm' (by default) for Black-Scholes-Merton, 'norm' for Bachelier

Returns volatility smile under the specified model

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class SabrChoiWu2021P(*sigma*, *vov=0.0*, *rho=0.0*, *beta=1.0*, *intr=0.0*, *divr=0.0*, *is_fwd=False*, *vol_beta=None*)

The CEV volatility approximation of the SABR modelbased on Theorem 2 of Choi & Wu (2019)

References

- Choi, J., & Wu, L. (2019). The equivalent constant-elasticity-of-variance (CEV) volatility of the stochastic-alpha-beta-rho (SABR) model. ArXiv:1911.13123 [q-Fin]. https://arxiv.org/abs/1911.13123

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.SabrChoiWu2021P(sigma=2, vov=0.2, rho=-0.3, beta=0.5)
>>> m.vol_for_price(np.arange(80, 121, 10), 100, 1.2)
array([2.07761123, 2.03665311, 2.00332   , 1.97718783, 1.95781579])
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([22.0470526 , 14.56114825,  8.74169054,  4.72447547,  2.29018838])
```

base_model(*vol*, *is_fwd=None*)

Create base model based on _base_beta value: *Norm* for 0, Cev for (0,1), and *Bsm* for 1 If *_base_beta* is None, use *base* instead.

Parameters

- **vol** – base model volatility
- **is_fwd** – if True, treat *spot* as forward price. False by default.

Returns: model

delta(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_benchmark(*set_no=None*)
Initiate a SABR model with stored benchmark parameter sets

Parameters **set_no** – set number

Returns Dataframe of all test cases if set_no = None (model, Dataframe of result, params) if set_no is specified

References

- Antonov, Alexander, Konikov, M., & Spector, M. (2013). SABR spreads its wings. Risk, 2013(Aug), 58–63.
- Antonov, Alexandre, Konikov, M., & Spector, M. (2019). Modern SABR Analytics. Springer International Publishing. https://doi.org/10.1007/978-3-030-10656-0
- Antonov, Alexandre, & Spector, M. (2012). Advanced analytics for the SABR model. Available at SSRN. https://ssrn.com/abstract=2026350
- Cai, N., Song, Y., & Chen, N. (2017). Exact Simulation of the SABR Model. Operations Research, 65(4), 931–951. https://doi.org/10.1287/opre.2017.1617
- Korn, R., & Tang, S. (2013). Exact analytical solution for the normal SABR model. Wilmott Magazine, 2013(7), 64–69. https://doi.org/10.1002/wilm.10235
- Lewis, A. L. (2016). Option valuation under stochastic volatility II: With Mathematica code. Finance Press.
- von Sydow, L., ..., Haentjens, T., & Waldén, J. (2018). BENCHOP - SLV: The BENCHmarking project in Option Pricing – Stochastic and Local Volatility problems. International Journal of Computer Mathematics, 1–14. https://doi.org/10.1080/00207160.2018.1544368

mass_zero(*spot*, *texp*, *log=False*)
Probability mass absorbed at the zero boundary (K=0)

Parameters

- **spot** – spot (or forward) price
- **texp** – time to expiry
- **log** – log value if True

Returns (log) probability mass at zero

mass_zero_t0(*spot*, *texp*)
Limit value of -T log(M_T) as T -> 0, where M_T is the mass at zero. See Corollary 3.1 of Choi & Wu (2019)

Parameters

- **spot** – spot (or forward) price
- **texp** – time to expiry

Returns -lim_{T->0} T log(M_T)

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_for_price(*strike*, *spot*, *texp*)
Equivalent volatility of the SABR model

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **texp** – time to expiry

Returns equivalent volatility

vol_from_mass_zero(*strike*, *spot*, *texp*, *mass=None*)
Implied volatility from positive mass at zero from DMHJ (2017) If mass is given, use the given value. If None (by default), compute model implied value.

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **mass** – probability mass at zero (None by default)

Returns implied BSM volatility

References

De Marco, S., Hillairet, C., & Jacquier, A. (2017). Shapes of Implied Volatility with Positive Mass at Zero. SIAM Journal on Financial Mathematics, 8(1), 709–737. https://doi.org/10.1137/14098065X

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model=None*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm'} 'bsm' (by default) for Black-Scholes-Merton, 'norm' for Bachelier

Returns volatility smile under the specified model

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class SabrHagan2002(*sigma*, *vov=0.0*, *rho=0.0*, *beta=1.0*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
SABR model with Hagan's implied volatility approximation for 0<beta<=1.

References

Hagan, P. S., Kumar, D., Lesniewski, A. S., & Woodward, D. E. (2002). Managing Smile Risk. Wilmott, September, 84–108.

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.SabrHagan2002(sigma=2, vov=0.2, rho=-0.3, beta=0.5)
>>> m.vol_for_price(np.arange(80, 121, 10), 100, 1.2)
array([0.21976016, 0.20922027, 0.200432  , 0.19311113, 0.18703486])
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([22.04862858, 14.56226187,  8.74170415,  4.72352155,  2.28891776])
```

base_model (*vol*, *is_fwd=None*)

Create base model based on _base_beta value: *Norm* for 0, Cev for (0,1), and *Bsm* for 1 If *_base_beta* is None, use *base* instead.

Parameters

- **vol** – base model volatility
- **is_fwd** – if True, treat *spot* as forward price. False by default.

Returns: model

calibrate3 (*price_or_vol3*, *strike3*, *spot*, *texp*, *cp=1*, *setval=False*, *is_vol=True*)

Given option prices or implied vols at 3 strikes, compute the sigma, vov, rho to fit the data using *scipy.optimize.root*. If prices are given (is_vol=False) convert the prices to vol first.

Parameters

- **price_or_vol3** – 3 prices or 3 volatilities (depending on *is_vol*)
- **strike3** – 3 strike prices
- **spot** – spot price
- **texp** – time to expiry
- **cp** – cp
- **setval** – if True, set sigma, vov, rho values
- **is_vol** – if True, *price_or_vol3* are volatilities.

Returns Dictionary of *sigma*, *vov*, and *rho*.

delta (*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_benchmark(*set_no=None*)
Initiate a SABR model with stored benchmark parameter sets

Parameters **set_no** – set number

Returns Dataframe of all test cases if set_no = None (model, Dataframe of result, params) if set_no is specified

References

- Antonov, Alexander, Konikov, M., & Spector, M. (2013). SABR spreads its wings. Risk, 2013(Aug), 58–63.
- Antonov, Alexandre, Konikov, M., & Spector, M. (2019). Modern SABR Analytics. Springer International Publishing. https://doi.org/10.1007/978-3-030-10656-0
- Antonov, Alexandre, & Spector, M. (2012). Advanced analytics for the SABR model. Available at SSRN. https://ssrn.com/abstract=2026350
- Cai, N., Song, Y., & Chen, N. (2017). Exact Simulation of the SABR Model. Operations Research, 65(4), 931–951. https://doi.org/10.1287/opre.2017.1617
- Korn, R., & Tang, S. (2013). Exact analytical solution for the normal SABR model. Wilmott Magazine, 2013(7), 64–69. https://doi.org/10.1002/wilm.10235
- Lewis, A. L. (2016). Option valuation under stochastic volatility II: With Mathematica code. Finance Press.
- von Sydow, L., ..., Haentjens, T., & Waldén, J. (2018). BENCHOP - SLV: The BENCHmarking project in Option Pricing – Stochastic and Local Volatility problems. International Journal of Computer Mathematics, 1–14. https://doi.org/10.1080/00207160.2018.1544368

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price

- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_for_price(*strike*, *spot*, *texp*)
Equivalent volatility of the SABR model

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **texp** – time to expiry

Returns equivalent volatility

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model=None*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm'} 'bsm' (by default) for Black-Scholes-Merton, 'norm' for Bachelier

Returns volatility smile under the specified model

volga (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class SabrLorig2017 (*sigma*, *vov=0.0*, *rho=0.0*, *beta=1.0*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Third-order BSM volatilty approximation of the SABR model by Lorig et al. (2017)

References

Lorig, M., Pagliarani, S., & Pascucci, A. (2017). Explicit Implied Volatilities for Multifactor Local-Stochastic Volatility Models. Mathematical Finance, 27(3), 926–960. https://doi.org/10.1111/mafi.12105

base_model (*vol*, *is_fwd=None*)
Create base model based on _base_beta value: *Norm* for 0, Cev for (0,1), and *Bsm* for 1 If *_base_beta* is None, use *base* instead.

Parameters

- **vol** – base model volatility
- **is_fwd** – if True, treat *spot* as forward price. False by default.

Returns: model

delta (*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_benchmark(*set_no=None*)
Initiate a SABR model with stored benchmark parameter sets

Parameters **set_no** – set number

Returns Dataframe of all test cases if set_no = None (model, Dataframe of result, params) if set_no is specified

References

- Antonov, Alexander, Konikov, M., & Spector, M. (2013). SABR spreads its wings. Risk, 2013(Aug), 58–63.
- Antonov, Alexandre, Konikov, M., & Spector, M. (2019). Modern SABR Analytics. Springer International Publishing. https://doi.org/10.1007/978-3-030-10656-0
- Antonov, Alexandre, & Spector, M. (2012). Advanced analytics for the SABR model. Available at SSRN. https://ssrn.com/abstract=2026350
- Cai, N., Song, Y., & Chen, N. (2017). Exact Simulation of the SABR Model. Operations Research, 65(4), 931–951. https://doi.org/10.1287/opre.2017.1617
- Korn, R., & Tang, S. (2013). Exact analytical solution for the normal SABR model. Wilmott Magazine, 2013(7), 64–69. https://doi.org/10.1002/wilm.10235
- Lewis, A. L. (2016). Option valuation under stochastic volatility II: With Mathematica code. Finance Press.
- von Sydow, L., ..., Haentjens, T., & Waldén, J. (2018). BENCHOP - SLV: The BENCHmarking project in Option Pricing – Stochastic and Local Volatility problems. International Journal of Computer Mathematics, 1–14. https://doi.org/10.1080/00207160.2018.1544368

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price

- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_for_price(*strike*, *spot*, *texp*)
Equivalent volatility of the SABR model

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **texp** – time to expiry

Returns equivalent volatility

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model=None*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm'} 'bsm' (by default) for Black-Scholes-Merton, 'norm' for Bachelier

Returns volatility smile under the specified model

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class SabrNormVolApprox(*sigma*, *vov=0.0*, *rho=0.0*, *beta=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*, *is_atmvol=False*)
Noram SABR model (beta=0) with normal volatility approximation.

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.SabrNormVolApprox(sigma=20, vov=0.8, rho=-0.3)
>>> m.vol_for_price(np.arange(80, 121, 10), 100, 1.2)
array([24.97568842, 22.78062691, 21.1072    , 20.38569729, 20.78963436])
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([23.70791426, 15.74437409,  9.22425529,  4.78754361,  2.38004685])
```

base_model(*vol*, *is_fwd=None*)
Create base model based on _base_beta value: *Norm* for 0, Cev for (0,1), and *Bsm* for 1 If *_base_beta* is None, use *base* instead.

Parameters

- **vol** – base model volatility
- **is_fwd** – if True, treat *spot* as forward price. False by default.

Returns: model

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_benchmark(*set_no=None*)
Initiate a SABR model with stored benchmark parameter sets

Parameters **set_no** – set number

Returns Dataframe of all test cases if set_no = None (model, Dataframe of result, params) if set_no is specified

References

- Antonov, Alexander, Konikov, M., & Spector, M. (2013). SABR spreads its wings. Risk, 2013(Aug), 58–63.
- Antonov, Alexandre, Konikov, M., & Spector, M. (2019). Modern SABR Analytics. Springer International Publishing. https://doi.org/10.1007/978-3-030-10656-0
- Antonov, Alexandre, & Spector, M. (2012). Advanced analytics for the SABR model. Available at SSRN. https://ssrn.com/abstract=2026350
- Cai, N., Song, Y., & Chen, N. (2017). Exact Simulation of the SABR Model. Operations Research, 65(4), 931–951. https://doi.org/10.1287/opre.2017.1617
- Korn, R., & Tang, S. (2013). Exact analytical solution for the normal SABR model. Wilmott Magazine, 2013(7), 64–69. https://doi.org/10.1002/wilm.10235
- Lewis, A. L. (2016). Option valuation under stochastic volatility II: With Mathematica code. Finance Press.
- von Sydow, L., …, Haentjens, T., & Waldén, J. (2018). BENCHOP - SLV: The BENCHmarking project in Option Pricing – Stochastic and Local Volatility problems. International Journal of Computer Mathematics, 1–14. https://doi.org/10.1080/00207160.2018.1544368

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry

- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_for_price(*strike*, *spot*, *texp*)
Equivalent volatility of the SABR model

Parameters

- **strike** – strike price
- **spot** – spot (or forward)
- **texp** – time to expiry

Returns equivalent volatility

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model=None*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm'} 'bsm' (by default) for Black-Scholes-Merton, 'norm' for Bachelier

Returns volatility smile under the specified model

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

5.6 SABR Model with Integration

class SabrUncorrChoiWu2021(*sigma*, *vov=0.0*, *rho=0.0*, *beta=1.0*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
The uncorrelated SABR (rho=0) model pricing by approximating the integrated variance with a log-normal distribution.

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> param = {"sigma": 0.4, "vov": 0.6, "rho": 0, "beta": 0.3, 'n_quad': 9}
>>> fwd, texp = 0.05, 1
>>> strike = np.array([0.4, 0.8, 1, 1.2, 1.6, 2.0]) * fwd
>>> m = pf.SabrUncorrChoiWu2021(**param)
>>> m.mass_zero(fwd, texp)
0.7623543217183134
>>> m.price(strike, fwd, texp)
array([0.04533777, 0.04095806, 0.03889591, 0.03692339, 0.03324944,
       0.02992918])
```

References

- Choi, J., & Wu, L. (2021). A note on the option price and `Mass at zero in the uncorrelated SABR model and implied volatility asymptotics'. Quantitative Finance (Forthcoming). https://doi.org/10.1080/14697688.2021.1876908
- Gulisashvili, A., Horvath, B., & Jacquier, A. (2018). Mass at zero in the uncorrelated SABR model and implied volatility asymptotics. Quantitative Finance, 18(10), 1753–1765. https://doi.org/10.1080/14697688.2018.1432883

static avgvar_lndist (*vovn*)
Lognormal distribution parameters of the normalized average variance: sigma^2 * texp * m1 * exp(sig*Z - 0.5*sig^2)

Parameters **vovn** – vov * sqrt(texp)

Returns (m1, sig) True distribution should be multiplied by sigma^2*t

5.7 Stochastic volatility with Fourier inversion

class BsmFft (*sigma*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Option pricing under Black-Scholes-Merton (BSM) model using fast fourier transform (FFT).

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.BsmFft(sigma=0.2, intr=0.05, divr=0.1)
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([15.71362027,  9.69251556,  5.52948647,  2.94558375,  1.4813909 ])
```

charfunc_logprice (*x*, *texp*)
Characteristic function of log price

Parameters

- **x** –
- **texp** –

Returns:

delta (*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

fft_interp(*texp*, **args*, ***kwargs*)
FFT method based on the Lewis expression

References

https://github.com/cantaro86/Financial-Models-Numerical-Methods/blob/master/1.3%20Fourier%20transform%20methods.ipynb

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

mgf_logprice(*uu*, *texp*)
Moment generating function (MGF) of log price. (forward = 1)

Parameters

- **xx** – dummy variable
- **texp** – time to expiry

Returns MGF value at xx

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.

- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class ExpNigFft(*sigma*, *vov=0.01*, *rho=0.0*, *mr=0.01*, *theta=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)

charfunc_logprice(*x*, *texp*)
Characteristic function of log price

Parameters

- **x** –
- **texp** –

Returns:

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

fft_interp(*texp*, **args*, ***kwargs*)
FFT method based on the Lewis expression

References

https://github.com/cantaro86/Financial-Models-Numerical-Methods/blob/master/1.3%20Fourier%20transform%20methods.ipynb

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_benchmark(*set_no=None*)
Initiate an SV model with stored benchmark parameter sets

Parameters **set_no** – set number

Returns Dataframe of all test cases if set_no = None (model, Dataframe of result, params) if set_no is specified

References:

mgf_logprice(*uu*, *texp*)

Parameters

- **uu** –
- **texp** –

Returns:

model_type
alias of `NotImplementedError`

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

var_process
alias of `NotImplementedError`

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model='bsm'*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm'} 'bsm' (by default) for Black-Scholes-Merton, 'norm' for Bachelier

Returns volatility smile under the specified model

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry

- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class HestonFft(*sigma*, *vov=0.01*, *rho=0.0*, *mr=0.01*, *theta=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Heston model option pricing with FFT

References

- Lewis AL (2000) Option valuation under stochastic volatility: with Mathematica code. Finance Press

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> strike = np.array([60, 70, 100, 140])
>>> sigma, vov, mr, rho, texp, spot = 0.04, 1, 0.5, -0.9, 10, 100
>>> m = pf.HestonFft(sigma, vov=vov, mr=mr, rho=rho)
>>> m.price(strike, spot, texp)
>>> # true price: 44.32997507, 35.8497697, 13.08467014, 0.29577444
array([44.32997507, 35.8497697 , 13.08467014,  0.29577444])
```

charfunc_logprice(*x*, *texp*)
Characteristic function of log price

Parameters

- **x** –
- **texp** –

Returns:

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

fft_interp(*texp*, **args*, ***kwargs*)
FFT method based on the Lewis expression

References

https://github.com/cantaro86/Financial-Models-Numerical-Methods/blob/master/1.3%20Fourier%20transform%20methods.ipynb

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_benchmark(*set_no=None*)
Initiate an SV model with stored benchmark parameter sets

Parameters **set_no** – set number

Returns Dataframe of all test cases if set_no = None (model, Dataframe of result, params) if set_no is specified

References:

mgf_logprice(*uu*, *texp*)
Log price MGF under the Heston model. We use the characteristic function in Eq (2.8) of Lord & Kahl (2010) that is continuous in branch cut when complex log is evaluated.

References

- Heston SL (1993) A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. The Review of Financial Studies 6:327–343. https://doi.org/10.1093/rfs/6.2.327
- Lord R, Kahl C (2010) Complex Logarithms in Heston-Like Models. Mathematical Finance 20:671–694. https://doi.org/10.1111/j.1467-9965.2010.00416.x

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

var_process
alias of NotImplementedError

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model='bsm'*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm'} 'bsm' (by default) for Black-Scholes-Merton, 'norm' for Bachelier

Returns volatility smile under the specified model

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class OusvFft(*sigma*, *vov=0.01*, *rho=0.0*, *mr=0.01*, *theta=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)

OUSV model option pricing with FFT

charfunc_logprice(*x*, *texp*)

Characteristic function of log price

Parameters

- **x** –
- **texp** –

Returns:

delta(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

fft_interp(*texp*, **args*, ***kwargs*)

FFT method based on the Lewis expression

References

https://github.com/cantaro86/Financial-Models-Numerical-Methods/blob/master/1.3%20Fourier%20transform%20methods.ipynb

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_benchmark(*set_no=None*)

Initiate an SV model with stored benchmark parameter sets

Parameters **set_no** – set number

Returns Dataframe of all test cases if set_no = None (model, Dataframe of result, params) if set_no is specified

References:

mgf_logprice(*uu*, *texp*)

Log price MGF under the OUSV model. We use the characteristic function in Eq (4.14) of Lord & Kahl (2010) that is continuous in branch cut when complex log is evaluated.

Returns MGF value at uu

References

- Lord R, Kahl C (2010) Complex Logarithms in Heston-Like Models. Mathematical Finance 20:671–694. https://doi.org/10.1111/j.1467-9965.2010.00416.x

mgf_logprice_schobelzhu1998(*uu*, *texp*)

MGF from Eq. (13) in Schobel & Zhu (1998). This form suffers discontinuity in complex log branch cut. Should not be used for pricing.

Parameters

- **uu** – dummy variable
- **texp** – time to expiry

Returns MGF value at uu

References

- Schöbel R, Zhu J (1999) Stochastic Volatility With an Ornstein–Uhlenbeck Process: An Extension. Rev Financ 3:23–46. https://doi.org/10.1023/A:1009803506170

params_kw()

Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)

Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

var_process
alias of `NotImplementedError`

vega (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_smile (*strike*, *spot*, *texp*, *cp=1*, *model='bsm'*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm'} 'bsm' (by default) for Black-Scholes-Merton, 'norm' for Bachelier

Returns volatility smile under the specified model

volga (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class VarGammaFft(*sigma*, *vov=0.01*, *rho=0.0*, *mr=0.01*, *theta=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)

charfunc_logprice(*x*, *texp*)
Characteristic function of log price

Parameters

- **x** –
- **texp** –

Returns:

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

fft_interp(*texp*, **args*, ***kwargs*)
FFT method based on the Lewis expression

References

https://github.com/cantaro86/Financial-Models-Numerical-Methods/blob/master/1.3%20Fourier%20transform%20methods.ipynb

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price

- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_benchmark(*set_no=None*)
Initiate an SV model with stored benchmark parameter sets

Parameters **set_no** – set number

Returns Dataframe of all test cases if set_no = None (model, Dataframe of result, params) if set_no is specified

References:

mgf_logprice(*uu*, *texp*)
Moment generating function (MGF) of log price. (forward = 1)

Parameters

- **xx** – dummy variable
- **texp** – time to expiry

Returns MGF value at xx

model_type
alias of `NotImplementedError`

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry

- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

var_process
alias of `NotImplementedError`

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vol_smile(*strike*, *spot*, *texp*, *cp=1*, *model='bsm'*)
Equivalent volatility smile for a given model

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option
- **model** – {'bsm', 'norm'} 'bsm' (by default) for Black-Scholes-Merton, 'norm' for Bachelier

Returns volatility smile under the specified model

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

5.8 Gamma distribution-related Models

class InvGam(*sigma*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Option pricing model with the inverse gamma (reciprocal gamma) distribution.

The parameters (alpha, beta) is from Wikipedia. https://en.wikipedia.org/wiki/Inverse-gamma_distribution Note that the n-th moment of the inverse gamma RV is beta^n / (alpha-1)*...*(alpha-n). Alpha and beta is calibrated to match the first two moments of the lognormal distribution with volatility sigma so that the option price is similar to that of the BSM model with volatility sigma.

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.InvGam(sigma=0.2, intr=0.05, divr=0.1)
>>> m.price(np.arange(80, 121, 10), 100, 1.2)
array([15.49803779,  9.53595458,  5.49889751,  3.02086661,  1.60505654])
```

alpha_beta(*spot*, *texp*)

Computes the inverse gamma distribution parameters (alpha, beta) from sigma, spot, texp.

m1 = beta/(alpha-1)

m2/m1^2 = exp(sigma^2 T) = (alpha-1)/(alpha-2)

Parameters

- **spot** – spot (or forward) price
- **texp** – time to expiry

Returns (alpha, beta)

price(*strike*, *spot*, *texp*, *cp=1*)

Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

5.9 Multiasset Models

class BsmBasket1Bm(*sigma*, *weight=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)

Multiasset BSM model for pricing basket/Spread options when all asset prices are driven by a single Brownian motion (BM).

delta(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) price.
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

static root(*fac*, *std*, *strike*)
Calculate the root x of f(x) = sum(fac * exp(std*x)) - strike = 0 using Newton's method

Each fac and std should have the same signs so that f(x) is a monotonically increasing function.

fac: factor to the exponents. (n_asset,) or (n_strike, n_asset). Asset takes the last dimension. std: total standard variance. (n_asset,) strike: strike prices. scalar or (n_asset,)

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class BsmBasketChoi2018(*sigma*, *cor=None*, *weight=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Choi (2018)'s pricing method for Basket/Spread/Asian options

References

- Choi J (2018) Sum of all Black-Scholes-Merton models: An efficient pricing method for spread, basket, and Asian options. Journal of Futures Markets 38:627–644. https://doi.org/10.1002/fut.21909

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

static householder(*vv0*)
Returns a Householder reflection (orthonormal matrix) that maps (1,0,...0) to vv0

Parameters **vv0** – vector

Returns Reflection matrix

References

- https://en.wikipedia.org/wiki/Householder_transformation

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price

- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_spread(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Initalize an instance for spread option pricing. This is a special case of the initalization with weight = (1, -1)

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.NormSpread.init_spread((20, 30), cor=-0.5, intr=0.05)
>>> m.price(np.arange(-2, 3) * 10, [100, 120], 1.3)
array([17.95676186, 13.74646821, 10.26669936,  7.47098719,  5.29057157])
```

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.

- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

v1_fwd_weight(*fwd*, *texp*)
Construct v1, forward array, and weights

Parameters

- **fwd** – forward vector of assets
- **texp** – time to expiry

Returns (v1, f_k, ww)

v_mat(*fwd*)
Construct the V matrix

Parameters **fwd** – forward vector of assets

Returns V matrix

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class BsmBasketJsu(*sigma*, *cor=None*, *weight=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)

Johnson's SU distribution approximation for Basket option pricing under the multiasset BSM model.

Note: Johnson's SU distribution is the solution of NSVh with NSVh with lambda = 1.

References

- Posner, S. E., & Milevsky, M. A. (1998). Valuing exotic options by approximating the SPD

with higher moments. The Journal of Financial Engineering, 7(2). https://ssrn.com/abstract=108539

- Choi, J., Liu, C., & Seo, B. K. (2019). Hyperbolic normal stochastic volatility model.

Journal of Futures Markets, 39(2), 186–204. https://doi.org/10.1002/fut.21967

delta(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)

Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)

Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)

Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_spread(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Initalize an instance for spread option pricing. This is a special case of the initalization with weight = (1, -1)

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.NormSpread.init_spread((20, 30), cor=-0.5, intr=0.05)
>>> m.price(np.arange(-2, 3) * 10, [100, 120], 1.3)
array([17.95676186, 13.74646821, 10.26669936,  7.47098719,  5.29057157])
```

moment_vsk(*fwd*, *texp*)
Return variance, skewness, kurtosis for Basket options.

Parameters

- **fwd** – forward price
- **texp** – time to expiry

Returns: variance, skewness, kurtosis of Basket options

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Basket options price. :param strike: strike price :param spot: spot price :param texp: time to expiry :param cp: 1/-1 for call/put option

Returns: Basket options price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class BsmBasketLevy1992(*sigma*, *cor=None*, *weight=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Basket option pricing with the log-normal approximation of Levy & Turnbull (1992)

References

- Levy E, Turnbull S (1992) Average intelligence. Risk 1992:53–57
- Krekel M, de Kock J, Korn R, Man T-K (2004) An analysis of pricing methods for basket options. Wilmott Magazine 2004:82–89

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> strike = np.arange(50, 151, 10)
>>> m = pf.BsmBasketLevy1992(sigma=0.4*np.ones(4), cor=0.5)
>>> m.price(strike, spot=100*np.ones(4), texp=5)
array([54.34281026, 47.521086  , 41.56701301, 36.3982413 , 31.92312156,
       28.05196621, 24.70229571, 21.800801  , 19.28360474, 17.09570196,
       15.19005654])
```

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_spread(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Initalize an instance for spread option pricing. This is a special case of the initalization with weight = (1, -1)

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.NormSpread.init_spread((20, 30), cor=-0.5, intr=0.05)
>>> m.price(np.arange(-2, 3) * 10, [100, 120], 1.3)
array([17.95676186, 13.74646821, 10.26669936,  7.47098719,  5.29057157])
```

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class BsmBasketMilevsky1998(*sigma*, *cor=None*, *weight=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Basket option pricing with the inverse gamma distribution of Milevsky & Posner (1998)

References

- Milevsky MA, Posner SE (1998) A Closed-Form Approximation for Valuing Basket Options. The Journal of Derivatives 5:54–61. https://doi.org/10.3905/jod.1998.408005
- Krekel M, de Kock J, Korn R, Man T-K (2004) An analysis of pricing methods for basket options. Wilmott Magazine 2004:82–89

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> strike = np.arange(50, 151, 10)
>>> m = pf.BsmBasketMilevsky1998(sigma=0.4*np.ones(4), cor=0.5)
>>> m.price(strike, spot=100*np.ones(4), texp=5)
array([51.93069524, 44.40986   , 38.02596564, 32.67653542, 28.21560931,
       24.49577509, 21.38543199, 18.77356434, 16.56909804, 14.69831445,
       13.10186928])
```

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_spread(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Initalize an instance for spread option pricing. This is a special case of the initalization with weight = (1, -1)

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.NormSpread.init_spread((20, 30), cor=-0.5, intr=0.05)
>>> m.price(np.arange(-2, 3) * 10, [100, 120], 1.3)
array([17.95676186, 13.74646821, 10.26669936,  7.47098719,  5.29057157])
```

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry

- **cp** – 1/-1 for call/put

Returns theta value

vanna (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class BsmMax2(*sigma*, *cor=None*, *weight=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Option on the max of two assets. Payout = max(max(F_1, F_2) - K, 0) for all or max(K - max(F_1, F_2), 0) for put option

References

- Rubinstein M (1991) Somewhere Over the Rainbow. Risk 1991:63–66

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.BsmMax2(0.2*np.ones(2), cor=0, divr=0.1, intr=0.05)
>>> m.price(strike=[90, 100, 110], spot=100*np.ones(2), texp=3)
array([15.86717049, 11.19568103,  7.71592217])
```

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry

- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class BsmSpreadBjerksund2014(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Bjerksund & Stensland (2014)'s approximation for spread option.

References

- Bjerksund P, Stensland G (2014) Closed form spread option valuation. Quantitative Finance 14:1785–1794. https://doi.org/10.1080/14697688.2011.617775

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.BsmSpreadBjerksund2014((0.2, 0.3), cor=-0.5)
>>> m.price(np.arange(-2, 3) * 10, [100, 120], 1.3)
array([22.13172022, 17.18304247, 12.98974214,  9.54431944,  6.80612597])
```

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class BsmSpreadKirk(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Kirk's approximation for spread option.

References

- Kirk E (1995) Correlation in the energy markets. In: Managing Energy Price Risk, First. Risk Publications, London, pp 71–78

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.BsmSpreadKirk((0.2, 0.3), cor=-0.5)
>>> m.price(np.arange(-2, 3) * 10, [100, 120], 1.3)
array([22.15632247, 17.18441817, 12.98974214,  9.64141666,  6.99942072])
```

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class NormBasket(*sigma*, *cor=None*, *weight=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Basket option pricing under the multiasset Bachelier model

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price

- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

classmethod init_spread(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)
Initalize an instance for spread option pricing. This is a special case of the initalization with weight = (1, -1)

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> m = pf.NormSpread.init_spread((20, 30), cor=-0.5, intr=0.05)
>>> m.price(np.arange(-2, 3) * 10, [100, 120], 1.3)
array([17.95676186, 13.74646821, 10.26669936,  7.47098719,  5.29057157])
```

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class OptMaABC(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *is_fwd=False*)

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry

- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

5.10 Multiasset Monte-Carlo Models

class BsmNdMc(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *rn_seed=None*, *antithetic=True*)
Monte-Carlo simulation of multiasset (N-d) BSM (geometric Brownian Motion)

Examples

```
>>> import pyfeng as pf
>>> spot = np.ones(4)*100
>>> sigma = np.ones(4)*0.4
>>> texp = 5
>>> payoff = lambda x: np.fmax(np.mean(x,axis=1) - strike, 0) # Basket option
>>> strikes = np.arange(80, 121, 10)
>>> m = pf.BsmNdMc(sigma, cor=0.5, rn_seed=1234)
>>> m.simulate(n_path=20000, tobs=[texp])
>>> p = []
>>> for strike in strikes:
>>>     p.append(m.price_european(spot, texp, payoff))
>>> np.array(p)
array([36.31612946, 31.80861014, 27.91269315, 24.55319506, 21.62677625])
```

delta(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price

- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward(*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

price_european(*spot*, *texp*, *payoff*)
The European price of that payoff at the expiry.

Parameters

- **spot** – array of spot prices
- **texp** – time-to-expiry
- **payoff** – payoff function applicable to the time-slice of price path

Returns The MC price of the payoff

simulate(*tobs*, *n_path*, *store=True*)
Simulate the price paths and store in the class. The initial prices are normalized to 0 and spot should be multiplied later.

Parameters

- **tobs** – array of observation times
- **n_path** – number of paths to simulate
- **store** – if True (default), store path, tobs, and n_path in the class

Returns price path (time, path, asset) if store is False

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

class NormNdMc(*sigma*, *cor=None*, *intr=0.0*, *divr=0.0*, *rn_seed=None*, *antithetic=True*)
Monte-Carlo simulation of multiasset (N-d) Normal/Bachelier model (arithmetic Brownian Motion)

Examples

```
>>> import pyfeng as pf
>>> spot = np.ones(4)*100
>>> sigma = np.ones(4)*0.4
>>> texp = 5
>>> payoff = lambda x: np.fmax(np.mean(x,axis=1) - strike, 0) # Basket option
>>> strikes = np.arange(80, 121, 10)
>>> m = pf.NormNdMc(sigma*spot, cor=0.5, rn_seed=1234)
>>> m.simulate(tobs=[texp], n_path=20000)
>>> p = []
>>> for strike in strikes:
>>>    p.append(m.price_european(spot, texp, payoff))
>>> np.array(p)
array([39.42304794, 33.60383167, 28.32667559, 23.60383167, 19.42304794])
```

delta (*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

delta_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model delta (sensitivity to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns delta value

forward (*spot*, *texp*)
Forward price

Parameters

- **spot** – spot price
- **texp** – time to expiry

Returns forward price

gamma (*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

gamma_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model gamma (2nd derivative to price) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns Delta with numerical derivative

impvol(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

impvol_brentq(*price*, *strike*, *spot*, *texp*, *cp=1*, *setval=False*)
Implied volatility using Brent's method. Slow but robust implementation.

Parameters

- **price** – option price
- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put
- **setval** – if True, sigma is set with the solved implied volatility

Returns implied volatility

params_kw()
Model parameters in dictionary

pdf_numeric(*strike*, *spot*, *texp*, *cp=- 1*, *h=0.001*)
Probability density functin (PDF) at *strike*

Parameters

- **strike** – strike price
- **spot** – spot price

- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns probability densitiy

price(*strike*, *spot*, *texp*, *cp=1*)
Call/put option price.

Parameters

- **strike** – strike price.
- **spot** – spot (or forward) prices for assets. Asset dimension should be the last, e.g. (n_asset,) or (N, n_asset)
- **texp** – time to expiry.
- **cp** – 1/-1 for call/put option.

Returns option price

price_european(*spot*, *texp*, *payoff*)
The European price of that payoff at the expiry.

Parameters

- **spot** – array of spot prices
- **texp** – time-to-expiry
- **payoff** – payoff function applicable to the time-slice of price path

Returns The MC price of the payoff

simulate(*tobs*, *n_path*, *store=True*)
Simulate the price paths and store in the class. The initial prices are normalized to 0 and spot should be added later.

Parameters

- **tobs** – array of observation times
- **n_path** – number of paths to simulate
- **store** – if True (default), store path, tobs, and n_path in the class

Returns price path (time, path, asset) if store is False

theta(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

theta_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model thegta (sensitivity to time-to-maturity) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns theta value

vanna(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vanna_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vanna (cross-derivative to price and volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns vanna value

vega(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

vega_numeric(*strike*, *spot*, *texp*, *cp=1*)
Option model vega (sensitivity to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot (or forward) price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put option

Returns vega value

volga (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

volga_numeric (*strike*, *spot*, *texp*, *cp=1*)
Option model volga (2nd derivative to volatility) by finite difference

Parameters

- **strike** – strike price
- **spot** – spot price
- **texp** – time to expiry
- **cp** – 1/-1 for call/put

Returns volga value

5.11 Asset Allocation Models

class RiskParity (*sigma=None*, *cor=None*, *cov=None*, *ret=None*, *budget=None*, *longshort=1*)
Risk parity (equal risk contribution) asset allocation.

References

- Maillard S, Roncalli T, Teïletche J (2010) The Properties of Equally Weighted Risk Contribution Portfolios. The Journal of Portfolio Management 36:60–70. https://doi.org/10.3905/jpm.2010.36.4.060
- Choi J, Chen R (2022) Improved iterative methods for solving risk parity portfolio. Journal of Derivatives and Quantitative Studies 30. https://doi.org/10.1108/JDQS-12-2021-0031

Examples

```
>>> import numpy as np
>>> import pyfeng as pf
>>> cov = np.array([
        [ 94.868, 33.750, 12.325, -1.178, 8.778 ],
        [ 33.750, 445.642, 98.955, -7.901, 84.954 ],
        [ 12.325, 98.955, 117.265, 0.503, 45.184 ],
        [ -1.178, -7.901, 0.503, 5.460, 1.057 ],
        [ 8.778, 84.954, 45.184, 1.057, 34.126 ]
    ])/10000
```

```
>>> m = pf.RiskParity(cov=cov)
>>> m.weight()
array([0.125, 0.047, 0.083, 0.613, 0.132])
>>> m._result
{'err': 2.2697290741335863e-07, 'n_iter': 6}
```

```
>>> m = pf.RiskParity(cov=cov, budget=[0.1, 0.1, 0.2, 0.3, 0.3])
>>> m.weight()
array([0.077, 0.025, 0.074, 0.648, 0.176])
```

```
>>> m = pf.RiskParity(cov=cov, longshort=[-1, -1, 1, 1, 1])
>>> m.weight()
array([-0.216, -0.162,  0.182,  0.726,  0.47 ])
```

classmethod init_random(*n_asset=10*, *zero_ev=0*, *budget=False*)

Randomly initialize the correlation matrix

Parameters

- **n_asset** – number of assets
- **zero_ev** – number of zero eivenvalues. 0 by default
- **budget** – randomize budget if True. False by default.

Returns RiskParity model object

weight(*tol=1e-06*)

Risk parity weight using the improved CCD method of Choi and Chen (2022)

Parameters **tol** – error tolerance

Returns risk parity weight

References

- Choi J, Chen R (2022) Improved iterative methods for solving risk parity portfolio. Journal of Derivatives and Quantitative Studies 30. https://doi.org/10.1108/JDQS-12-2021-0031

weight_ccd_original(*tol=1e-06*)

Risk parity weight using original CCD method of Griveau-Billion et al (2013). This is implemented for performance comparison. Use weight() for better performance.

Parameters **tol** – error tolerance

Returns risk parity weight

References

- Griveau-Billion T, Richard J-C, Roncalli T (2013) A Fast Algorithm for Computing High-dimensional Risk Parity Portfolios. arXiv:13114057 [q-fin]

weight_newton(*tol=1e-06*)

Risk parity weight using the 'improved' Newton method by Choi & Chen (2022). This is implemented for performance comparison. Use weight() for better performance.

Parameters **tol** – error tolerance

Returns risk parity weight

References

- Spinu F (2013) An Algorithm for Computing Risk Parity Weights. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.2297383
- Choi J, Chen R (2022) Improved iterative methods for solving risk parity portfolio. Journal of Derivatives and Quantitative Studies 30. https://doi.org/10.1108/JDQS-12-2021-0031

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search