Fourier Transforms

yellapantula.av

December 2020

1 Fourier Transforms

Recall that a Fourier Transform (FT) is a mathematical function that decomposes functions with space or time dependent variables depending on spacial or temporal frequencies [citation for def of FT]. Since our BSM and Heston models are time based, its apt to use FT here. We must know the probability distribution for the random variable within the BSM and Heston models. Thus, the characteristic function must be formed.

Let us recall the definition of the characteristic function $\phi_X(\cdot)$ of the random variable X:

$$\begin{aligned}\phi_X(u) &= \mathbb{E}\big[e^{iuX}\big] \\ &= \int_{\mathbb{R}} e^{iux} f_X(x)dx.\end{aligned} \tag{1}$$

where f_X is the density of X. The characteristic function is nothing but the Fourier transform of f_X.
Let's go over Fourier Transform properties:

1. $\phi_X(u)$ always exists. This is because $\left|\int_{\mathbb{R}} e^{iux} f_X(x)dx\right| \leq \int_{\mathbb{R}} |e^{iux} f_X(x)|dx < \infty$ Recall that $|e^{iux}| = 1$ and $\int_{\mathbb{R}} |f_X(x)|dx = 1$
2. $\phi_X(0) = 1$
3. $\phi_X(u)^* = \phi_X(-u)$, where $*$ means complex conjugate.

The density function can be obtained by taking the inverse Fourier transform:

$$f_X(x) = \frac{1}{2\pi}\int_{\mathbb{R}} e^{-iux}\phi_X(u)du$$

1.1 Inversion Theorem

Let us consider the following representation of the cumulative density function:

$$\begin{aligned}F_X(x) = \mathbb{P}(X < x) &= \int_{-\infty}^{x} f_X(t)dt \\ &= \frac{1}{2} - \frac{1}{2\pi}\int_{\mathbb{R}} \frac{e^{-iux}\phi_X(u)}{iu}du\end{aligned}$$

By taking the derivative of F_X, you can check that it gives the expression for f_X.
See the proof in the appendix:

1.2 Gil Peleaz

$$Re[z] = \frac{z + z^*}{2} \qquad Im[z] = \frac{z - z^*}{i2}$$

Starting from the inversion formula, Gil Pelaez [5] obtained the following expression:

$$\begin{aligned}
F_X(x) &= \frac{1}{2} - \frac{1}{2\pi}\int_{\mathbb{R}} e^{-iux}\phi_X(u)\cdot\frac{1}{iu}\,du \\
&= \frac{1}{2} - \frac{1}{2\pi}\int_{-\infty}^{0} e^{-iux}\phi_X(u)\frac{1}{iu}\,du + \frac{1}{2\pi}\int_{0}^{\infty} e^{-iux}\phi_X(u)\frac{1}{iu}\,du \\
&= \frac{1}{2} - \frac{1}{2\pi}\int_{0}^{\infty}\Big(-e^{iux}\phi_X(-u) + e^{-iux}\phi_X(u)\Big)\frac{1}{iu}\,du \\
&= \frac{1}{2} - \frac{1}{\pi}\int_{0}^{\infty}\frac{Im[e^{-iux}\phi_X(u)]}{u}du
\end{aligned}$$

This formula can also be written as:

$$F_X(x) = \frac{1}{2} - \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-iux}\phi_X(u)}{iu}\right]du$$

We obtain the expression for the density by taking the derivative:

$$f_X(x) = \frac{1}{\pi}\int_0^\infty Re\Big[e^{-iux}\phi_X(u)\Big]du$$

1.3 Pricing by Fourier Inversion

we found that the pricing formula for a call option with stike K and maturity T is:

$$C(S_t, K, T) = S_t\tilde{\mathbb{Q}}(S_T > K) - e^{-r(T-t)}K\,\mathbb{Q}(S_T > K)$$

where $\tilde{\mathbb{Q}}$ is the probability under the stock numeraire and $\mathbb{Q}$ is the probability under the money market numeraire.

This formula is model independent

When the stock follows a geometric Brownian motion, this formula corresponds to the Black-Scholes formula.

Now let us express $\tilde{\mathbb{Q}}$ and $\mathbb{Q}$ in terms of the characteristic function using the Gil Pelaez formula. Let us call $k = \log\frac{K}{S_t}$ and $S_T = S_t e^{X_T}$

For $\mathbb{Q}$ we can write:

$$\begin{aligned}
\mathbb{Q}(S_T > K) &= 1 - \mathbb{Q}(S_T < K) = 1 - \mathbb{Q}(X_T < k) \\
&= \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-iuk}\phi_X(u)}{iu}\right]du
\end{aligned}$$

In the same way, for $\tilde{\mathbb{Q}}$, we can write:

$$\begin{aligned}
\tilde{\mathbb{Q}}(S_T > K) &= \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-iuk}\tilde{\phi}_X(u)}{iu}\right]du \\
&= \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-iuk}\phi_X(u-i)}{iu\,\phi_X(-i)}\right]du
\end{aligned}$$

where:

$$\begin{aligned}
\tilde{\phi}_X(u) &= \mathbb{E}^{\tilde{\mathbb{Q}}}\left[e^{iuX_T}\right] = \mathbb{E}^{\mathbb{Q}}\left[\frac{d\tilde{\mathbb{Q}}}{d\mathbb{Q}}e^{iuX_T}\right] \\
&= \mathbb{E}^{\mathbb{Q}}\left[\frac{S_t e^{X_T}}{S_t\mathbb{E}^{\mathbb{Q}}[e^{X_T}]}e^{iuX_T}\right] \\
&= \mathbb{E}^{\mathbb{Q}}\left[\frac{e^{(iu+1)X_T}}{\phi_X(-i)}\right] \\
&= \frac{\phi_X(u-i)}{\phi_X(-i)}
\end{aligned}$$

1.4 Normal, Gamma, Poisson

Normal

```
def cf_normal(u,mu=1,sig=2):
    return np.exp(1j*u*mu-0.5*u**2*sig**2)
```

$\mathcal{N}(\mu, \sigma^2)$, $\sigma > 0$.

$$\phi_N(u) = e^{i\mu u - \frac{1}{2}\sigma^2 u^2}$$

Gamma

```
def cf_gamma(u,a=1,b=2):
    return (1-b*u*1j)**(-a)
```

Shape Scale Parametrization:
$\Gamma(a, b)$, $a, b > 0$.

$$\phi_G(u) = (1 - ibu)^{-a}$$

Poisson

```
def cf_poisson(u,lam=1):
    return np.exp(lam*(np.exp(1j*u)-1))
```

$Po(\lambda)$, $\lambda > 0$.

$$\phi_P(u) = e^{\lambda(e^{iu}-1)}$$

In [1]:

```
# !pip install functions
# !pip install nbconvert
# !export PATH=/Library/TeX/texbin:$PATH
import os
os.environ['PATH'].split(';')
# !pip install pdflatex
```

Out[1]:

```
['/home/avocado/anaconda3/bin:/home/avocado/anaconda3/condabin:/home/avocado/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin']
```

In [17]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.stats as ss
from scipy.integrate import quad
from functools import partial
from scipy.fftpack import fft, ifft
from scipy.interpolate import interp1d

# from functions.BS_pricer import BS_pricer
# from functions.Parameters import Option_param
# from functions.Processes import Merton_process
# from functions.Merton_pricer import Merton_pricer
# from functions.Processes import VG_process
# from functions.VG_pricer import VG_pricer
```

In [11]:

```
def cf_normal(u,mu=1,sig=2):
    return np.exp(1j*u*mu-0.5*u**2*sig**2)
```

In [12]:

```
def cf_gamma(u,a=1,b=2):
    return (1-b*u*1j)**(-a)
```

In [13]:

```
def cf_poisson(u,lam=1):
    return np.exp(lam*(np.exp(1j*u)-1))
```

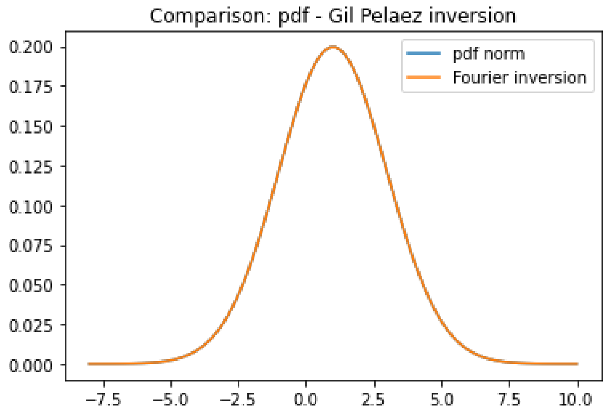
In [14]:

```
# Giz Peleaz
# right_lim is the right extreme of integration
# cf is the characteristic function
def Gil_Pelaez_pdf(x, cf, right_lim):
    integrand = lambda u: np.real( np.exp(-u*x*1j) * cf(u) )
    return 1/np.pi * quad(integrand, 1e-15, right_lim )[0]
```

Normal

In [18]:

```
x = np.linspace(-8,10,100)
plt.plot(x,ss.norm.pdf(x, loc=1, scale=2), label="pdf norm")
plt.plot(x,[Gil_Pelaez_pdf(i,cf_normal,np.inf) for i in x], label="Fourier inversion" )
plt.title("Comparison: pdf - Gil Pelaez inversion"); plt.legend()
plt.show()
```

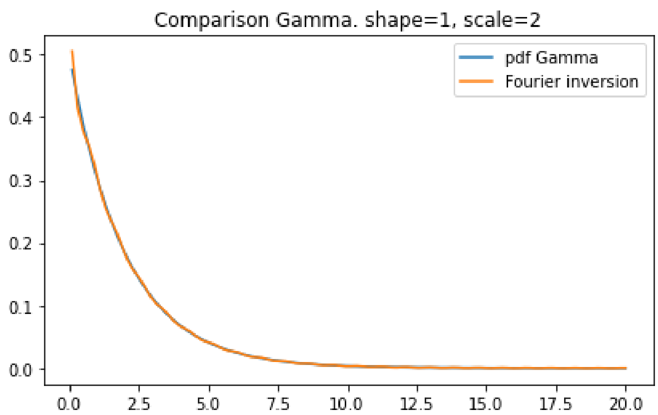


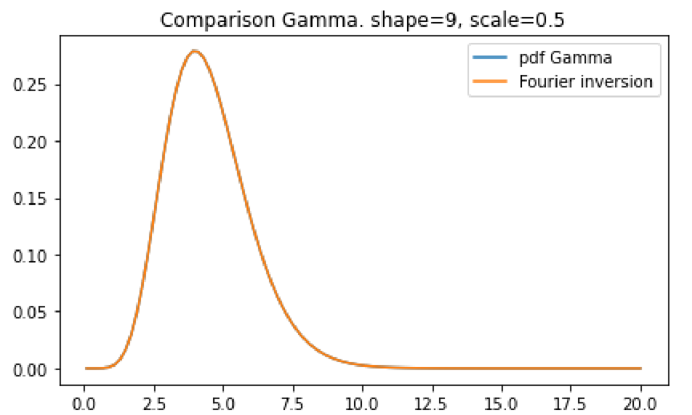
Gamma

In [19]:

```
xx = np.linspace(0.1,20,100)
a = 1     #shape parameter
b = 2     #scale parameter
c = 9     #shape parameter
d = 0.5   #scale parameter
lim_ab = 24
lim_cd = np.inf
cf_gamma_ab = partial(cf_gamma, a=a, b=b)   # function binding
cf_gamma_cd = partial(cf_gamma, a=c, b=d)   # function binding

fig = plt.figure(figsize=(15,4))
ax1 = fig.add_subplot(121); ax2 = fig.add_subplot(122)
ax1.plot(xx,ss.gamma.pdf(xx, a, scale=b), label="pdf Gamma")
ax1.plot(xx,[Gil_Pelaez_pdf(i,cf_gamma_ab, lim_ab) for i in xx], label="Fourier inversion" )
ax1.set_title("Comparison Gamma. shape=1, scale=2"); ax1.legend()
ax2.plot(xx,ss.gamma.pdf(xx,c, scale=d), label="pdf Gamma")
ax2.plot(xx,[Gil_Pelaez_pdf(i,cf_gamma_cd, lim_cd) for i in xx], label="Fourier inversion" )
ax2.set_title("Comparison Gamma. shape=9, scale=0.5"); ax2.legend()
plt.show()
```



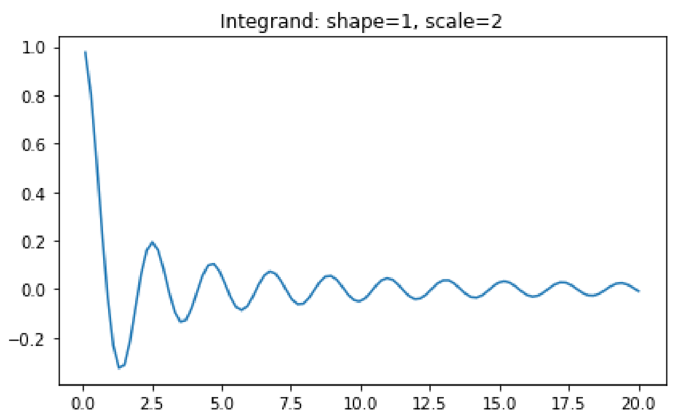


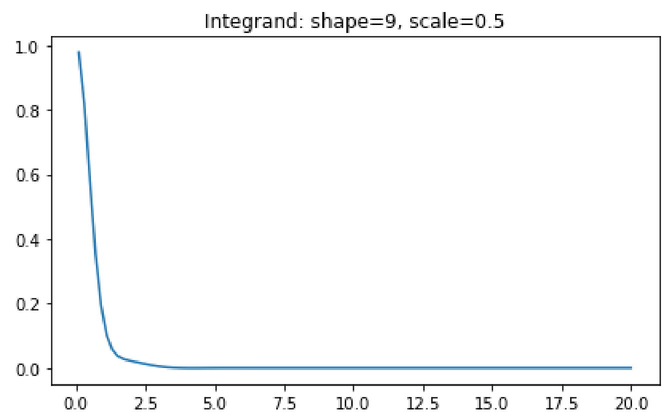
In [20]:

```
# Let's have a look at the plot of the integrands: (at the point x=3)

u = np.linspace(0.1,20,100)
x = 3
f = lambda u: np.real( np.exp(-u*x*1j) * cf_gamma_ab(u) )  # integrand
g = lambda u: np.real( np.exp(-u*x*1j) * cf_gamma_cd(u) )

fig = plt.figure(figsize=(15,4)); ax1 = fig.add_subplot(121); ax2 = fig.add_subplot(122)
ax1.plot(u, f(u)); ax1.set_title("Integrand: shape=1, scale=2")
ax2.plot(u, g(u)); ax2.set_title("Integrand: shape=9, scale=0.5")
plt.show()
```





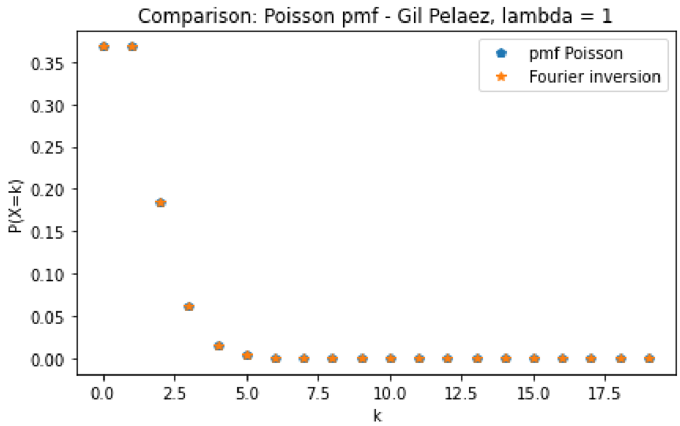
Poisson

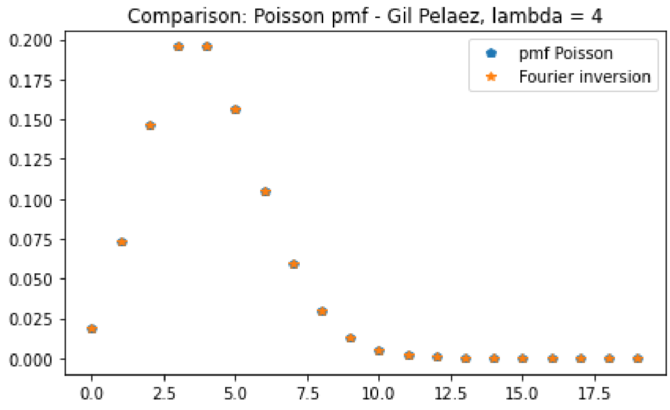
In [21]:

```
k = np.array(range(20))
lam = 4
cf_poisson4 = partial(cf_poisson, lam=lam)   # function binding to lam=4

fig = plt.figure(figsize=(15,4))
ax1 = fig.add_subplot(121); ax2 = fig.add_subplot(122)
ax1.plot(k, ss.poisson.pmf(k, 1), linestyle="None", marker='p', label="pmf Poisson")  # with lam=1
ax1.plot(k, [Gil_Pelaez_pdf(i,cf_poisson, np.pi) for i in k], \
```

```
        linestyle="None", marker='*', label="Fourier inversion" )      # lam=1 by default
ax1.set_xlabel("k"); ax1.set_ylabel("P(X=k)")
ax1.set_title("Comparison: Poisson pmf - Gil Pelaez, lambda = 1"); ax1.legend()
ax2.plot(k, ss.poisson.pmf(k, lam), linestyle="None", marker='p', label="pmf Poisson")
ax2.plot(k, [Gil_Pelaez_pdf(i,cf_poisson4, np.pi) for i in k], \
        linestyle="None", marker='*', label="Fourier inversion" )
ax2.set_title("Comparison: Poisson pmf - Gil Pelaez, lambda = 4"); ax2.legend()
plt.show()
```





In [ ]: