Brownian Motion Simulation - Brownian Paths

In [3]:
```
import numpy as np
```

In [5]:
```
import matplotlib.pyplot as plt
```

We set the random seed, for reproducibility.

In [7]:
```
np.random.seed(1821)
```

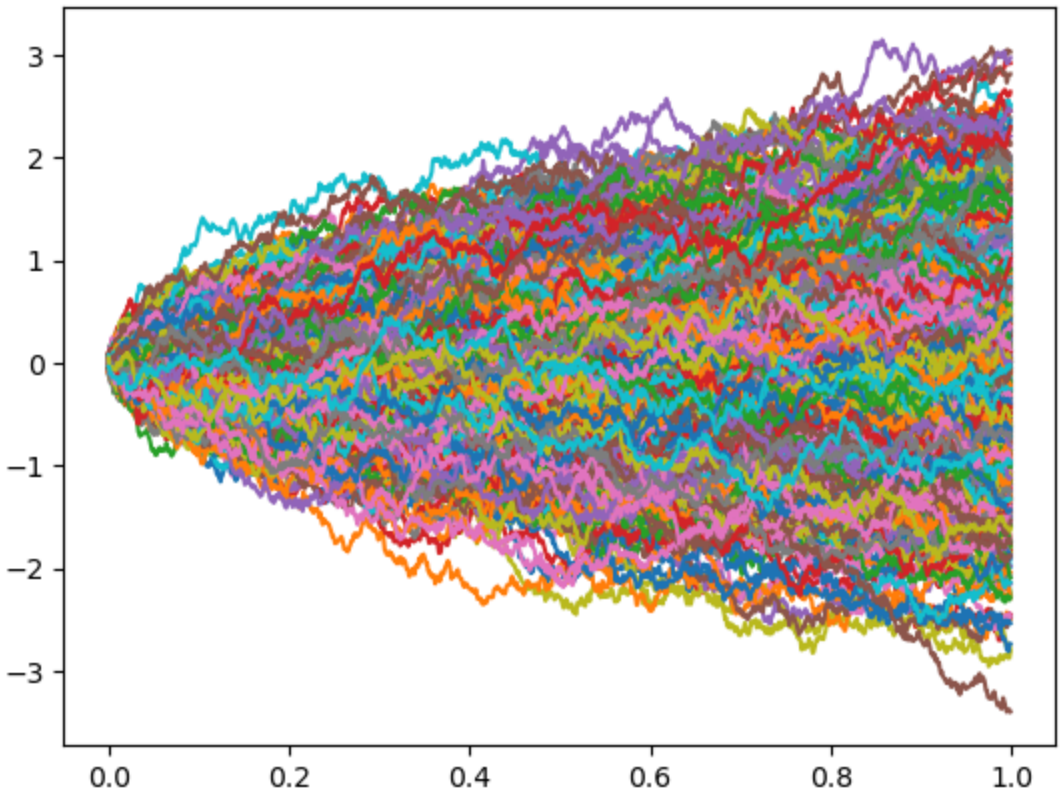
We now set the parameters for the Brownian motion. The time increment is decided to be $dt = 0.001$. The `timesteps` matrix is created using the `np.arange` command. The random number themselves by using the `np.random.randn` command, while the brownian trajectories are computed by accumulating along each individual path by using the `np.cumsum` command.

In [9]:
```
dt = 0.001
timesteps = np.arange(0, 1, dt)
n_paths = 5_000
```

In [10]:
```
rands = np.random.randn(len(timesteps), n_paths)
```

In [12]:
```
brownian = np.cumsum(rands, axis = 0) * np.sqrt(dt)
```

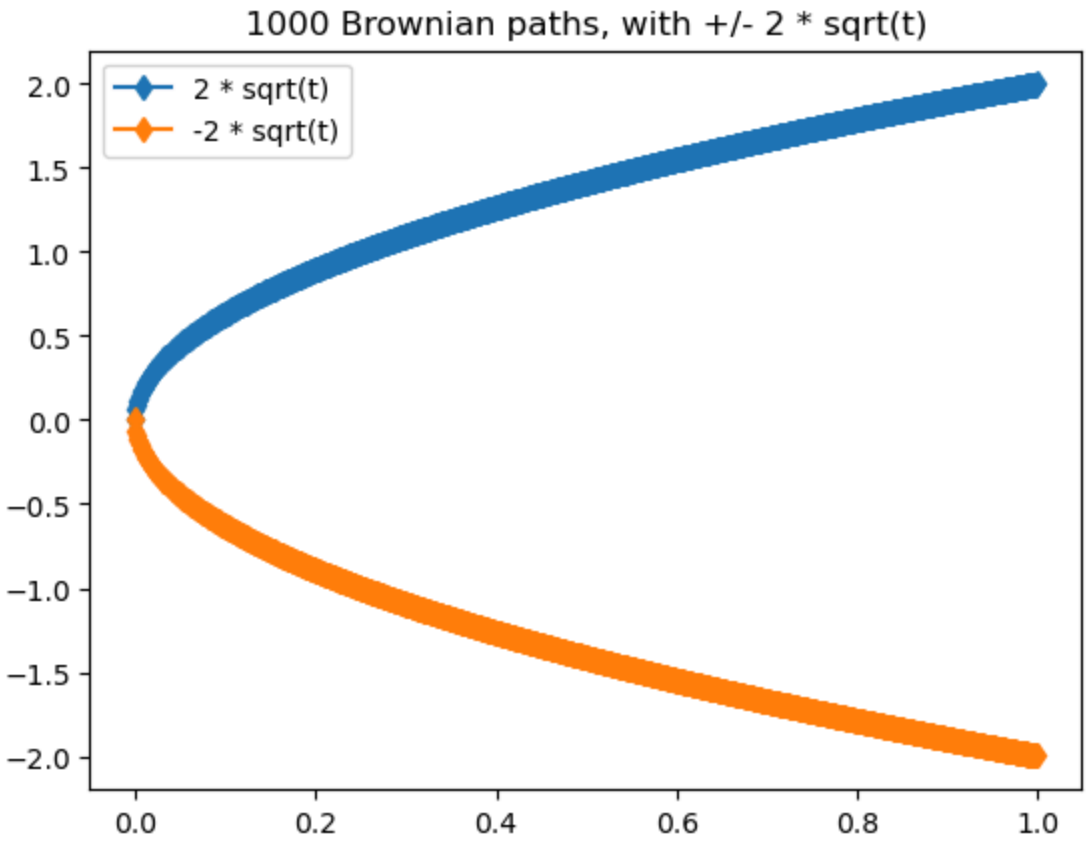
In [19]:
```
for i in range(1_000):
    plt.plot(timesteps, brownian[:, i])
```



After modelling the trajectories of the process, we plot the boundaries into which 95.4% of the motions will fall. We find this through the normal distribution, as Wiener processes are normally distributed with $\mu = 0$ and $\sigma = 1$. So we plot the paths which fall within $\pm 2 \times \sigma$.

In [21]:
```
plt.plot(timesteps, 2 * np.sqrt(timesteps), 'd-', label = '2 * sqrt(t)')
plt.plot(timesteps, -2 * np.sqrt(timesteps), 'd-', label = '-2 * sqrt(t)')

plt.title('1000 Brownian paths, with +/- 2 * sqrt(t)')
plt.legend()
plt.show()
```


1000 Brownian paths, with +/- 2 * sqrt(t)
2 * sqrt(t)
-2 * sqrt(t)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0