Réduire la dimension en préservant les distances
Partie numerique

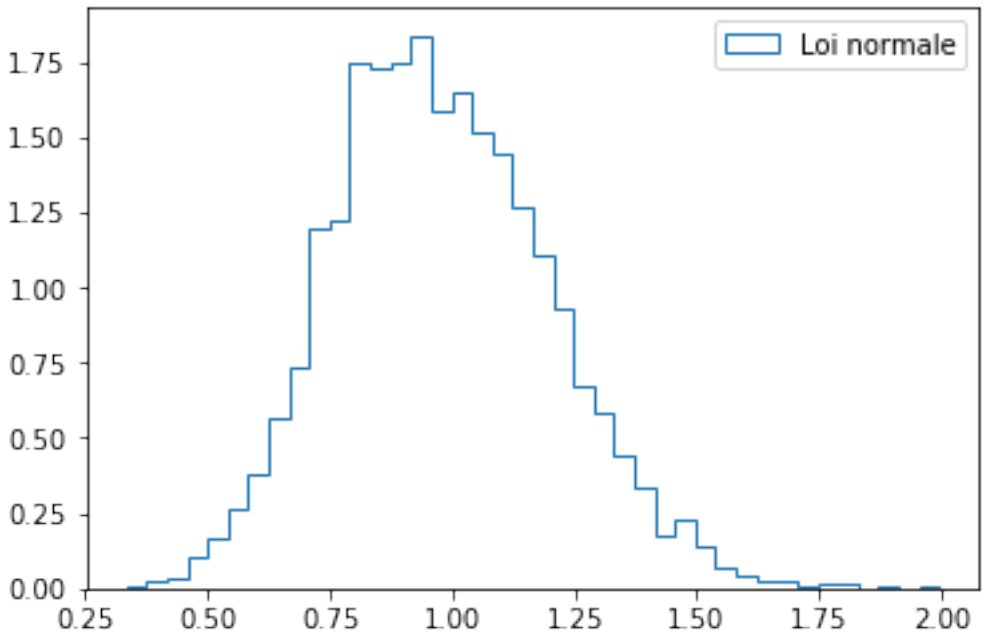
Gross Viktoriia, Sinitambirivoutin Maïdie

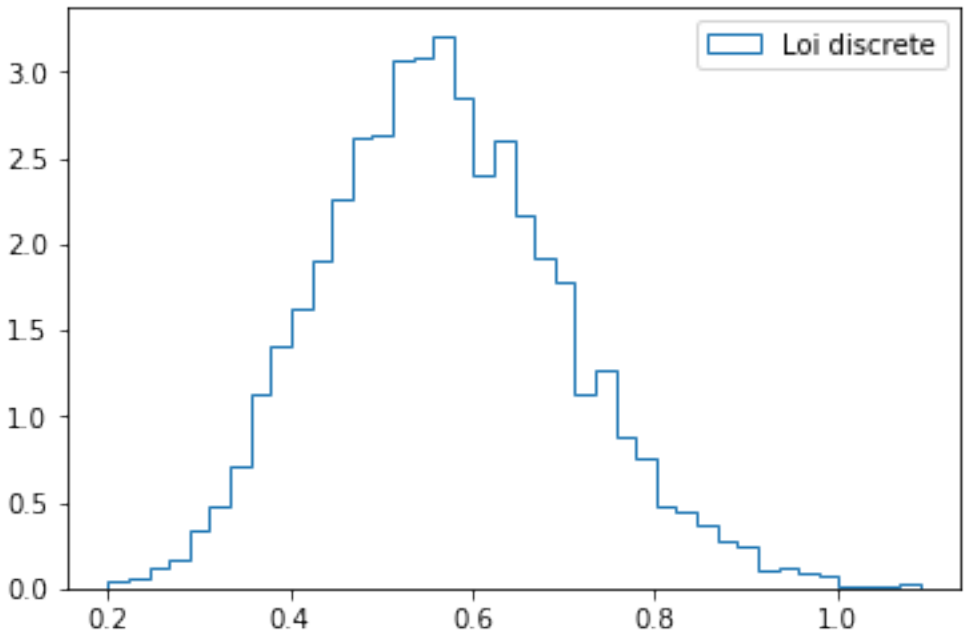
June 28, 2018

2. Deuxième cas est plus simple à calculer car il est discrete et pas continue.
3. Realisons les histogrammes des valeurs $\frac{||\tilde{x}_i - \tilde{x}_j||_2}{||x_i - x_j||_2}$ pour differents n, d, D.

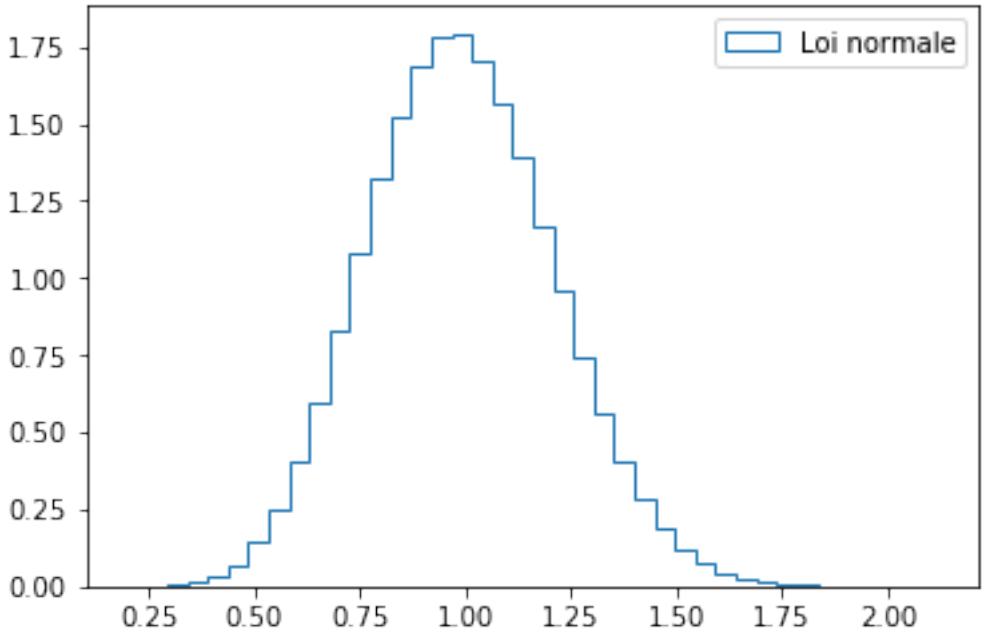
In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps

n = 100
D = 1000
d = 10
sigma = 1

X = np.random.rand(n, D)
A1 = 1/np.sqrt(d) * np.random.randn(D, d)

Y = np.matmul(X, A1)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi normale")
plt.legend(loc='best')
plt.show()


A2 = 1/np.sqrt(d) * np.random.choice(np.array([-1, 0, 0, 0, 0, 1]), (D, d))

Y = np.matmul(X, A2)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi discrete")
```

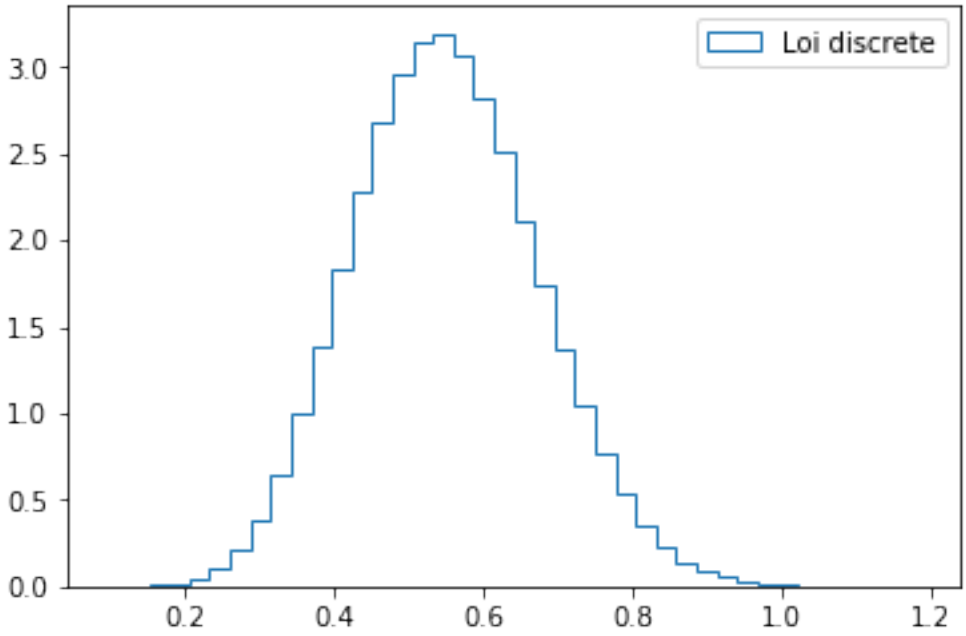
```
plt.legend(loc='best')
plt.show()
```





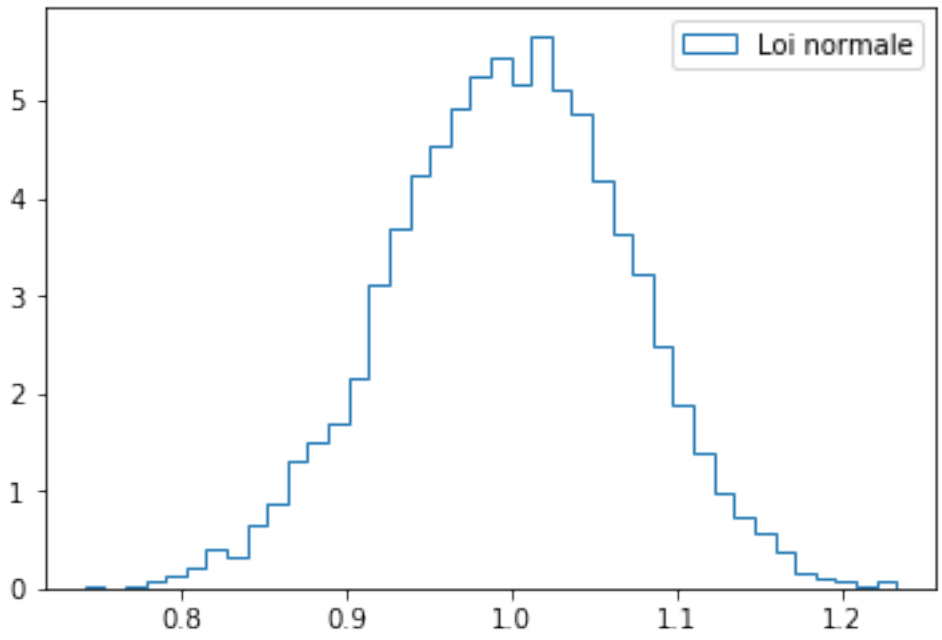
In [4]:
```
n = 1000
D = 1000
d = 10
sigma = 1

X = np.random.rand(n, D)
A1 = 1/np.sqrt(d) * np.random.randn(D, d)

Y = np.matmul(X, A1)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi normale")
plt.legend(loc='best')
plt.show()


A2 = 1/np.sqrt(d) * np.random.choice(np.array([-1, 0, 0, 0, 0, 1]), (D, d))

Y = np.matmul(X, A2)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi discrete")
plt.legend(loc='best')
plt.show()
```

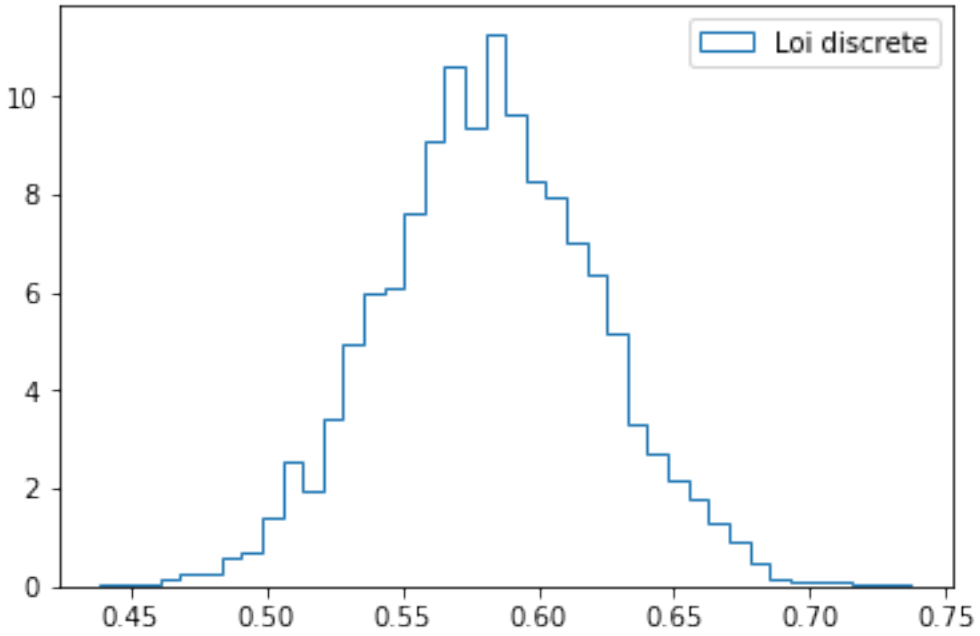




In [5]:
```
n = 100
D = 1000
d = 100
```

```
sigma = 1

X = np.random.rand(n, D)
A1 = 1/np.sqrt(d) * np.random.randn(D, d)

Y = np.matmul(X, A1)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi normale")
plt.legend(loc='best')
plt.show()


A2 = 1/np.sqrt(d) * np.random.choice(np.array([-1, 0, 0, 0, 0, 1]), (D, d))

Y = np.matmul(X, A2)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi discrete")
plt.legend(loc='best')
plt.show()
```



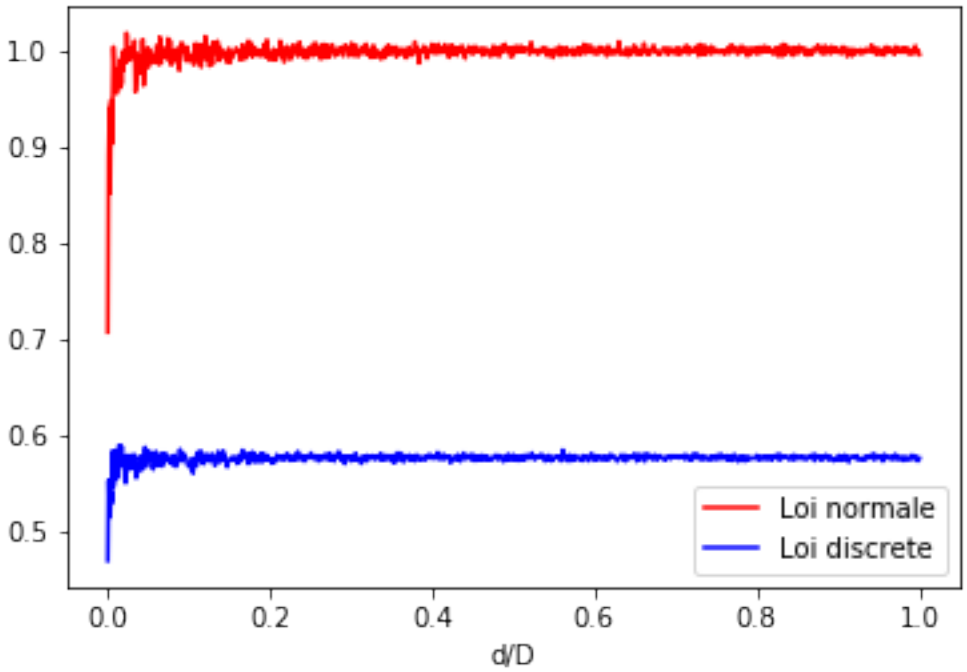


Nous pouvons voir que avec augmentation de n la forme de histogramme ressemble plus une gaussien. Avec augmentation de d/D le moyen converge vers 1 pour le cas de gaussien et vers 0,6 pour le cas de discrete. Il semble que dispersion diminue.

4. Resumons les resultats en graphique represantant la dispersion des histogrammes autour de 1 selon la valeur d/D.

```
In [8]: import numpy as np
        import matplotlib.pyplot as plt
        import scipy.stats as sps

        def mean1(d, D, n):
            X = np.random.rand(n, D)
            A1 = 1/np.sqrt(d) * np.random.randn(D, d)

            Y = np.matmul(X, A1)
            Z = []
            for i in range (n):
                for j in range (i + 1, n):
                    Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))
            return np.mean(Z)

        def mean2(d, D, n):
            X = np.random.rand(n, D)
            A2 = 1/np.sqrt(d) * np.random.choice(np.array([-1, 0, 0, 0, 0, 1]), (D, d))
```

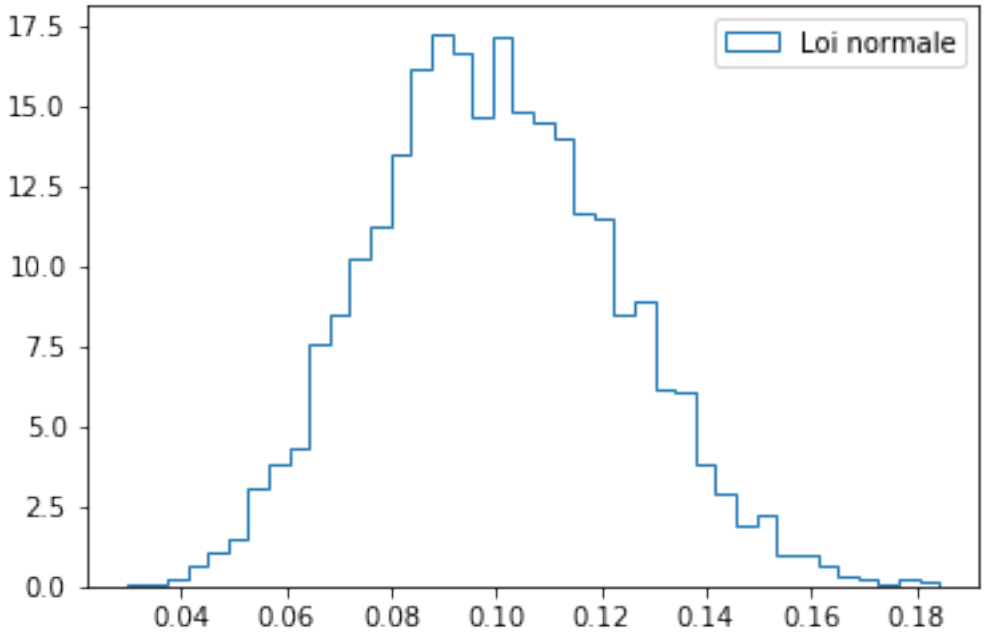
```
    Y = np.matmul(X, A2)
    Z = []
    for i in range (n):
        for j in range (i + 1, n):
            Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))

    return np.mean(Z)

n = 100
D = 1000
d = np.linspace(1, D, 1000)
Var1 = [mean1(int(i), D, n) for i in d]
Var2 = [mean2(int(i), D, n) for i in d]


plt.plot(d / D,Var1,'r',label="Loi normale")
plt.plot(d / D, Var2,"b",label="Loi discrete")
plt.xlabel("d/D")
plt.legend(loc='best')
plt.show()
```

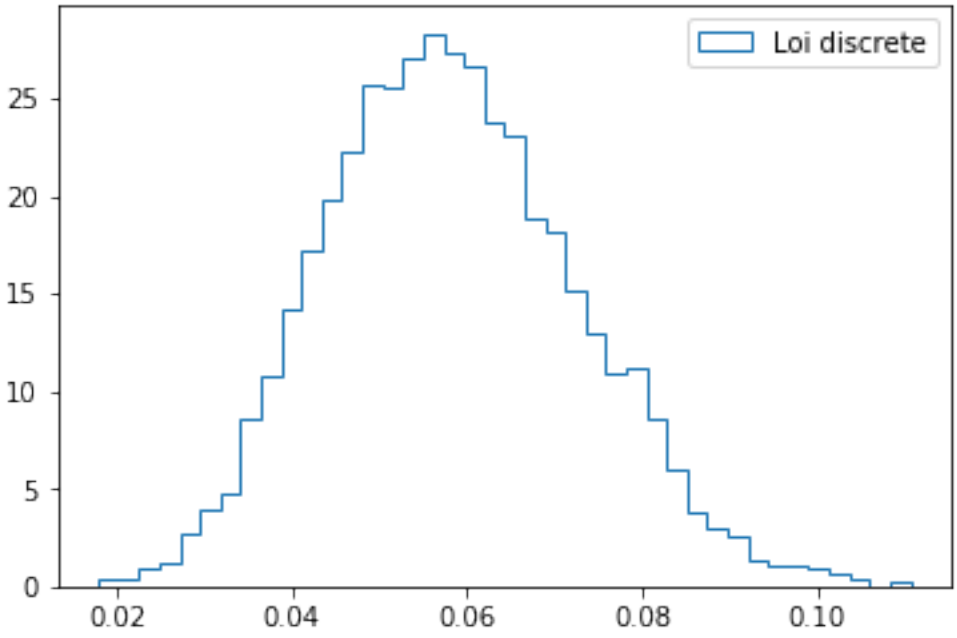


Nous pouvons deduire que pour le cas de gaussien le moyen converge vers 1 et à partir de certain ratio distance entre les vecteurs reste les même que avant la reduction des dimensions. Pour le cas discrete le moyen converge vers 0,6.

5. Realisons les mêmes expériences avec d'autres normes que la norme euclidienne: $max(\sum |x_i|)$ et $max(|x_i|)$.

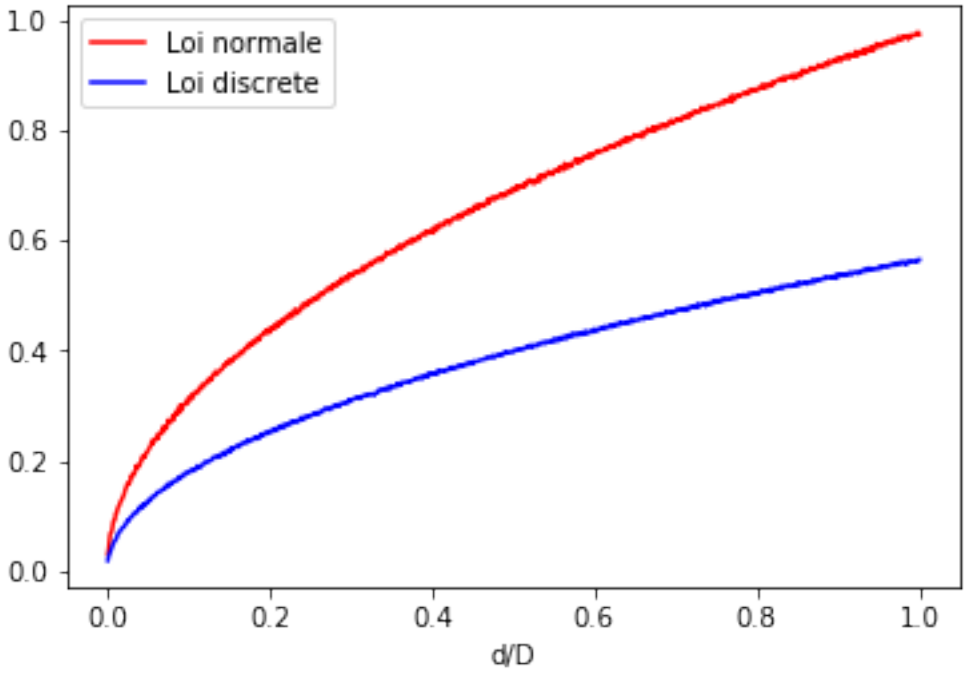
In [9]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps

n = 100
D = 1000
d = 10
sigma = 1

X = np.random.rand(n, D)
A1 = 1/np.sqrt(d) * np.random.randn(D, d)

Y = np.matmul(X, A1)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j], 1)  / np.linalg.norm(X[i] - X[j], 1))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi normale")
plt.legend(loc='best')
plt.show()


A2 = 1/np.sqrt(d) * np.random.choice(np.array([-1, 0, 0, 0, 0, 1]), (D, d))

Y = np.matmul(X, A2)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j], 1)  / np.linalg.norm(X[i] - X[j], 1))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi discrete")
plt.legend(loc='best')
plt.show()
```






```
In [11]: import numpy as np
         import matplotlib.pyplot as plt
         import scipy.stats as sps
```

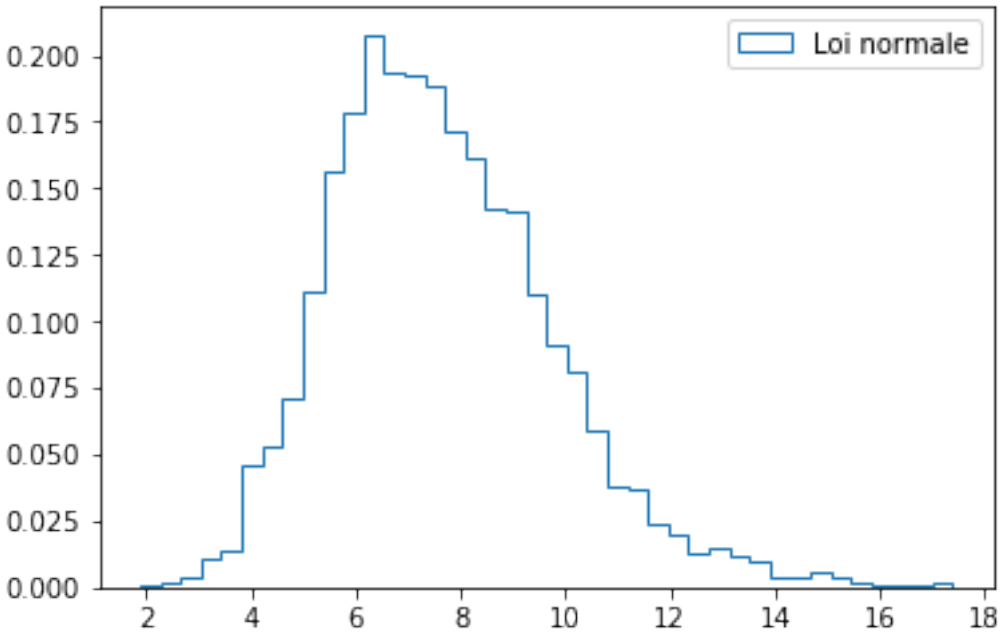
```
def mean1(d, D, n):
    X = np.random.rand(n, D)
    A1 = 1/np.sqrt(d) * np.random.randn(D, d)

    Y = np.matmul(X, A1)
    Z = []
    for i in range (n):
        for j in range (i + 1, n):
            Z.append(np.linalg.norm(Y[i] - Y[j], 1)  / np.linalg.norm(X[i] - X[j], 1))
    return np.mean(Z)

def mean2(d, D, n):
    X = np.random.rand(n, D)
    A2 = 1/np.sqrt(d) * np.random.choice(np.array([-1, 0, 0, 0, 0, 1]), (D, d))

    Y = np.matmul(X, A2)
    Z = []
    for i in range (n):
        for j in range (i + 1, n):
            Z.append(np.linalg.norm(Y[i] - Y[j], 1)  / np.linalg.norm(X[i] - X[j], 1))

    return np.mean(Z)

n = 100
D = 1000
d = np.linspace(1, D, 1000)
Var1 = [mean1(int(i), D, n) for i in d]
Var2 = [mean2(int(i), D, n) for i in d]

plt.plot(d / D,Var1,'r',label="Loi normale")
plt.plot(d / D, Var2,"b",label="Loi discrete")
plt.xlabel("d/D")
plt.legend(loc='best')
plt.show()
```

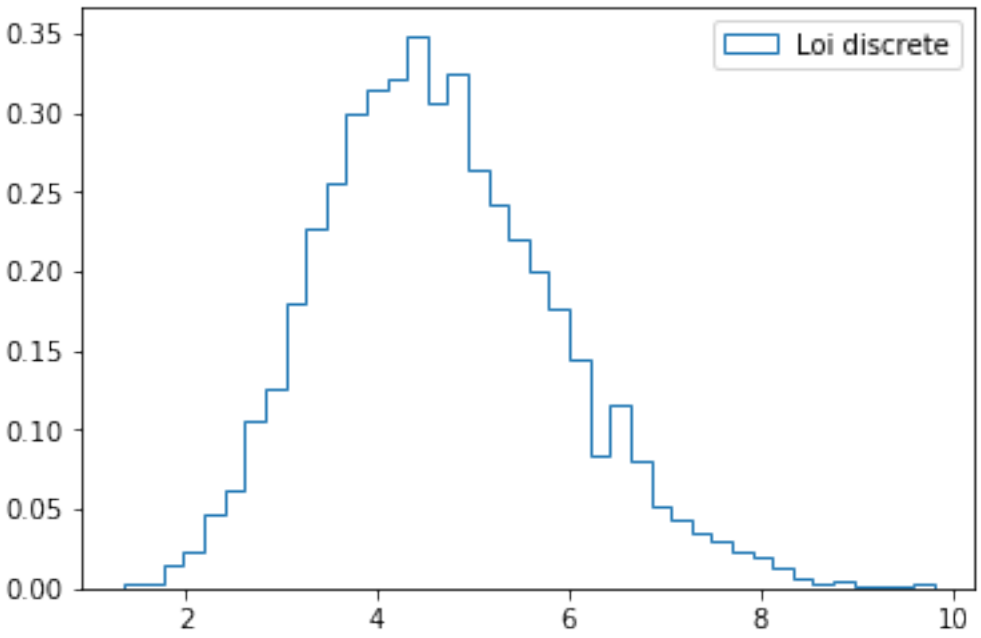


In [12]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps

n = 100
D = 1000
d = 10
sigma = 1

X = np.random.rand(n, D)
A1 = 1/np.sqrt(d) * np.random.randn(D, d)

Y = np.matmul(X, A1)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j], np.inf)  / np.linalg.norm(X[i] - X[j], np.

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi normale")
plt.legend(loc='best')
plt.show()


A2 = 1/np.sqrt(d) * np.random.choice(np.array([-1, 0, 0, 0, 0, 1]), (D, d))
```

```
Y = np.matmul(X, A2)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j], np.inf)  / np.linalg.norm(X[i] - X[j], np.
```

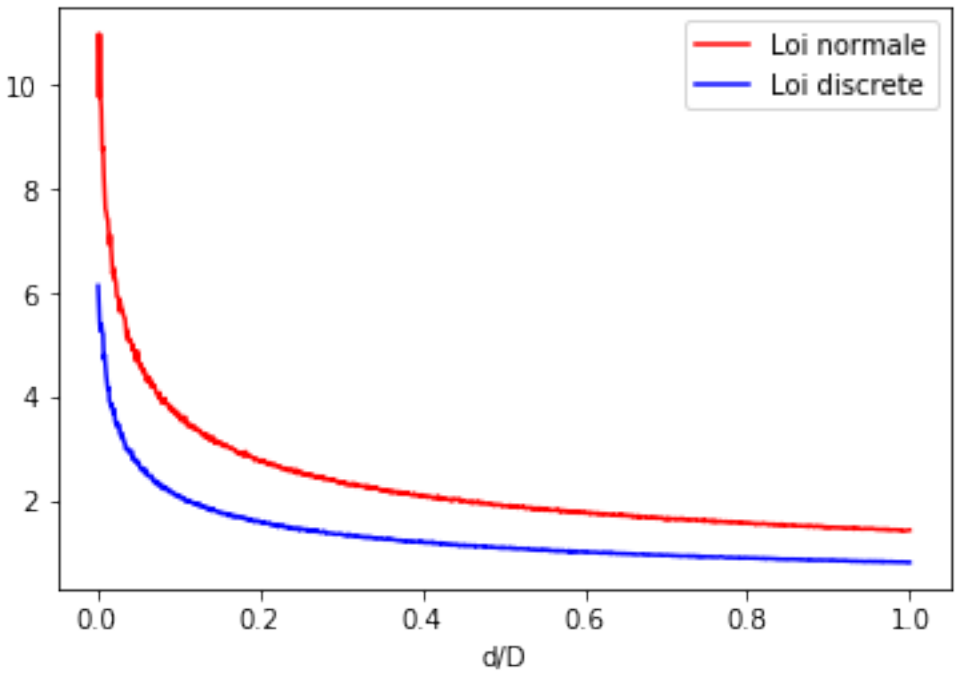
```
plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi discrete")
plt.legend(loc='best')
plt.show()
```





In [14]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps

def mean1(d, D, n):
    X = np.random.rand(n, D)
    A1 = 1/np.sqrt(d) * np.random.randn(D, d)

    Y = np.matmul(X, A1)
    Z = []
    for i in range (n):
        for j in range (i + 1, n):
            Z.append(np.linalg.norm(Y[i] - Y[j], np.inf)  / np.linalg.norm(X[i] - X[j],
    return np.mean(Z)

def mean2(d, D, n):
    X = np.random.rand(n, D)
    A2 = 1/np.sqrt(d) * np.random.choice(np.array([-1, 0, 0, 0, 0, 1]), (D, d))

    Y = np.matmul(X, A2)
    Z = []
    for i in range (n):
        for j in range (i + 1, n):
            Z.append(np.linalg.norm(Y[i] - Y[j], np.inf)  / np.linalg.norm(X[i] - X[j],

    return np.mean(Z)
```

```
n = 100
D = 1000
d = np.linspace(1, D, 1000)
Var1 = [mean1(int(i), D, n) for i in d]
Var2 = [mean2(int(i), D, n) for i in d]

plt.plot(d / D,Var1,'r',label="Loi normale")
plt.plot(d / D, Var2,"b",label="Loi discrete")
 plt.xlabel("d/D")
plt.legend(loc='best')
plt.show()
```

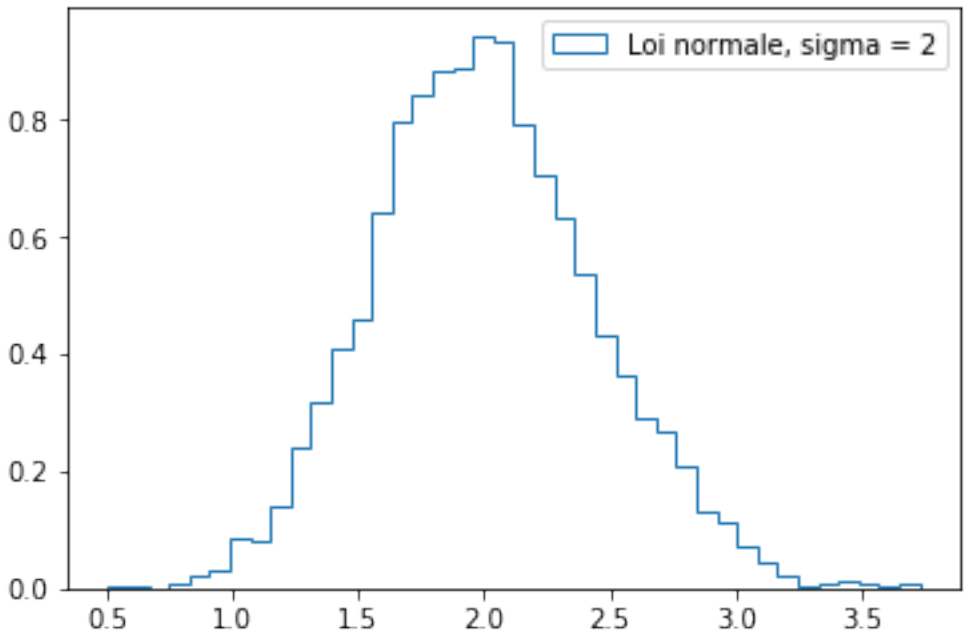


Dans tous les normes les moyens converges vers 1 pour le loi normal et vers 0,6 pour le loi discrete. Pour les normes 1 et 2 cette convergence est monotonne et croissante. Pour la norme $max|x_i|$ convergence est decroissante.

6. Utilison l'autre valeur σ pour le cas gaussien.

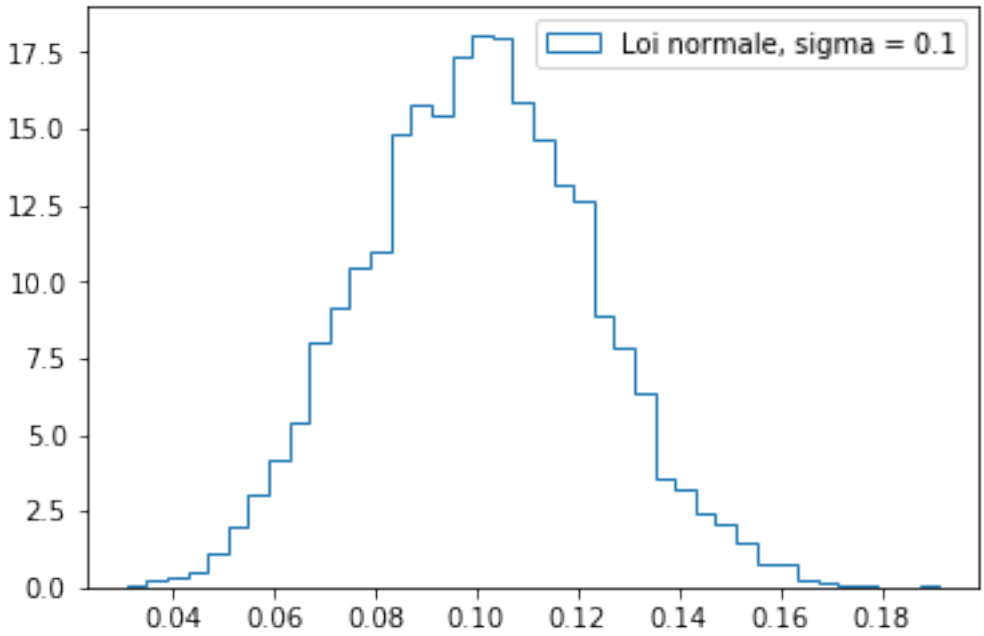
```
In [3]: import numpy as np
        import matplotlib.pyplot as plt
        import scipy.stats as sps

        n = 100
        D = 1000
        d = 10
        sigma = 2
```

```
X = np.random.rand(n, D)
A1 = 1/np.sqrt(d) * np.random.normal(0, sigma, (D, d))

Y = np.matmul(X, A1)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi normale, sigma = 2")
plt.legend(loc='best')
plt.show()
```



In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps

n = 100
D = 1000
d = 10
sigma = 0.1

X = np.random.rand(n, D)
A1 = 1/np.sqrt(d) * np.random.normal(0, sigma, (D, d))
```

```
Y = np.matmul(X, A1)
Z = []
for i in range (n):
    for j in range (i + 1, n):
        Z.append(np.linalg.norm(Y[i] - Y[j])  / np.linalg.norm(X[i] - X[j]))

plt.hist(Z, bins=40,normed=1,histtype='step',label="Loi normale, sigma = 0.1")
plt.legend(loc='best')
plt.show()
```



Avec changement de σ chanhe la moyen. Elle devient proche à valeur de σ.