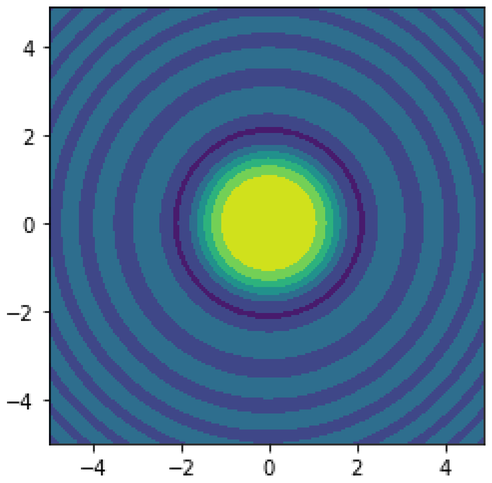
In [34]:

```python
import numpy as np
na, nb = (5, 3)

a = np. linspace(1, 2, na)

b = np. linspace(1, 2, nb)

xa, xb = np. meshgrid(a, b)

print('XA VALUES: \n', xa)

print('XB VALUES: \n', xb)
```

```
XA VALUES: 
 [[1.   1.25 1.5  1.75 2.  ]
 [1.   1.25 1.5  1.75 2.  ]
 [1.   1.25 1.5  1.75 2.  ]]
XB VALUES: 
 [[1.  1.  1.  1.  1. ]
 [1.5 1.5 1.5 1.5 1.5]
 [2.  2.  2.  2.  2. ]]
```

In [6]:

```python
import matplotlib.pyplot as plt

x = np.arange(-5, 5, 0.1)

y = np.arange(-5, 5, 0.1)

xx, yy = np.meshgrid(x, y, sparse=True)

z = np.sin(xx**2 + yy**2) / (xx**2 + yy**2)

h = plt.contourf(x, y, z)

plt.axis('scaled')

plt.show()
```



In [15]:
```
import numpy as np

import matplotlib.pyplot as plt

nx, ny = (3, 2)

x = np.linspace(0, 1, nx)

y = np.linspace(0, 1, ny)

xv, yv = np.meshgrid(x, y)


print('XV Values: ',xv)

print('YV Values: ',yv)
```

```
XV Values:  [[0.  0.5 1. ]
 [0.  0.5 1. ]]
YV Values:  [[0. 0. 0.]
 [1. 1. 1.]]
```

In [17]:
```
xv, yv = np.meshgrid(x, y, sparse=True)  # make sparse output arrays

print('XV Values: ',xv)

print('YV Values: ',yv)
```

```
XV Values:  [[0.  0.5 1. ]]
YV Values:  [[0.]
 [1.]]
```

In [24]:

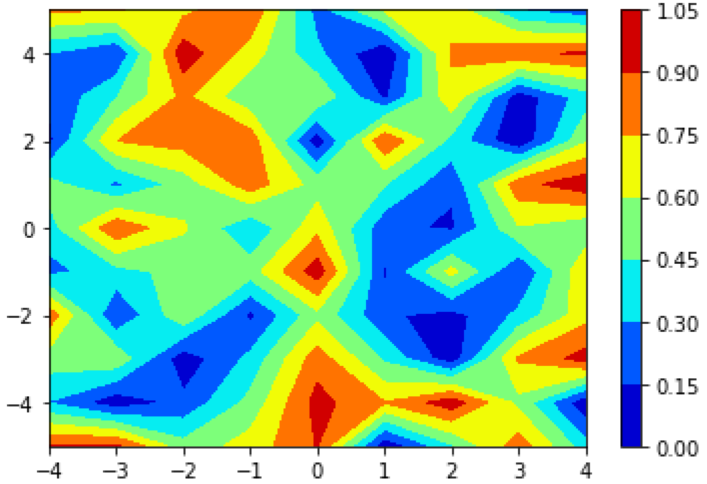
```
# Sample code for generation of first example
import numpy as np

import matplotlib.pyplot as plt
# from matplotlib import pyplot as plt
# pyplot imported for plotting graphs

x = np.linspace(-4, 4, 9)

# numpy.linspace creates an array of
# 9 linearly placed elements between
# -4 and 4, both inclusive
y = np.linspace(-5, 5, 11)

# The meshgrid function returns
# two 2-dimensional arrays
x_1, y_1 = np.meshgrid(x, y)

print("x_1 = ")
print(x_1)

print("y_1 = ")
print(y_1)
```

```
x_1 =
[[-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]]
y_1 =
[[-5. -5. -5. -5. -5. -5. -5. -5. -5.]
 [-4. -4. -4. -4. -4. -4. -4. -4. -4.]
 [-3. -3. -3. -3. -3. -3. -3. -3. -3.]
 [-2. -2. -2. -2. -2. -2. -2. -2. -2.]
 [-1. -1. -1. -1. -1. -1. -1. -1. -1.]
```

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.]
[ 2.  2.  2.  2.  2.  2.  2.  2.  2.]
[ 3.  3.  3.  3.  3.  3.  3.  3.  3.]
[ 4.  4.  4.  4.  4.  4.  4.  4.  4.]
[ 5.  5.  5.  5.  5.  5.  5.  5.  5.]]
```

In [27]:

```python
import numpy as np

import matplotlib.pyplot as plt

random_data = np.random.random((11, 9))
plt.contourf(x_1, y_1, random_data, cmap = 'jet')

plt.colorbar()

plt.show()
```

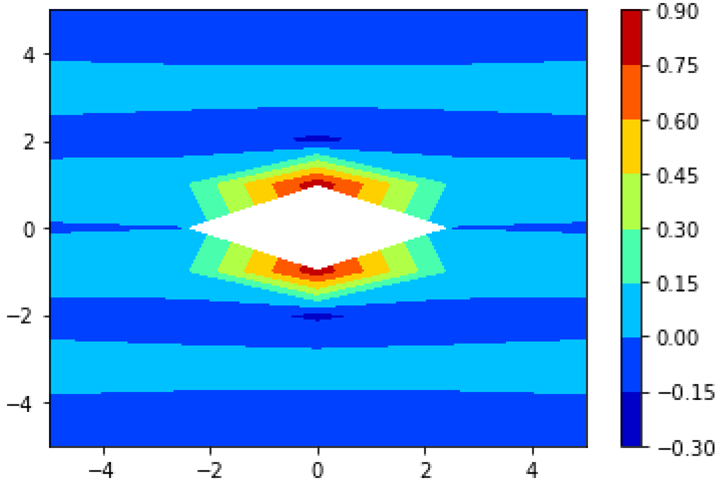


In [35]:

```python
import numpy as np

import matplotlib.pyplot as plt

a = np.linspace(-5, 5, 5)

b = np.linspace(-5, 5, 11)

random_data = np.random.random((11, 5))
```

```python
xa, xb = np.meshgrid(a, b)

sine = (np.sin(xa**2 + xb**2))/(xa**2 + xb**2)

plt.contourf(xa, xb, sine, cmap = 'jet')

plt.colorbar()

plt.show()
```

```
<ipython-input-35-e8f3e5881618>:13: RuntimeWarning: invalid value encountered in true_divide
  sine = (np.sin(xa**2 + xb**2))/(xa**2 + xb**2)
```



In [12]:
```python
import numpy as np
np.linspace(2.0, 10.0, num=5, retstep=True, endpoint = True)
```

Out[12]:
```
(array([ 2.,  4.,  6.,  8., 10.]), 2.0)
```

In [37]:
```python
#Draw samples from the distribution:
mu, sigma = 0, 0.1 # mean and standard deviation

s = np.random.normal(mu, sigma, 1000)

abs(mu - np.mean(s))
```

Out[37]:
```
0.002583663202036574
```

In [38]:
```
abs(sigma - np.std(s, ddof=1))
```

Out[38]: 0.00035673693922083664

In [39]:
```
#Display the histogram of the samples, along with the probability density function:
import matplotlib.pyplot as plt

count, bins, ignored = plt.hist(s, 30, density=True)

plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) *
               np.exp( - (bins - mu)**2 / (2 * sigma**2) ), linewidth=2, color='r')

plt.show()
```