In [1]:

```
# Import PySwarms
import numpy as np
import pyswarms as ps
from pyswarms.utils.functions import single_obj as fx
import random
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
```

In [2]:

```
# ... I also made some experiments with PySwarm
```

In [3]:

```
# nest: (0.2, 0.5)
# target: (0.9, 0.5)
```

In [4]:

```
# Adapted from: https://machinelearningmastery.com/a-gentle-introduction-to-part
```

In [ ]:

```

```

In [5]:

```
#n_particles = 10
#X = np.random.rand(2, n_particles)
#V = np.random.randn(2, n_particles)
```

In [6]:

```
#n_particles = 3
#print(np.random.rand(2, n_particles)*0.1 + 0.2)
```

In [7]:

```
n_particles = 3
print(np.random.rand(2, n_particles)*0.1 + 0.2)
print(np.random.rand(2, n_particles)*0.1 + 0.5)
```

```
[[0.2729313  0.26242812 0.29452105]
 [0.24524105 0.28515993 0.26575323]]
[[0.51176126 0.55173374 0.53757823]
 [0.53421638 0.57985235 0.58254328]]
```

In [ ]:

```

```

In [ ]:

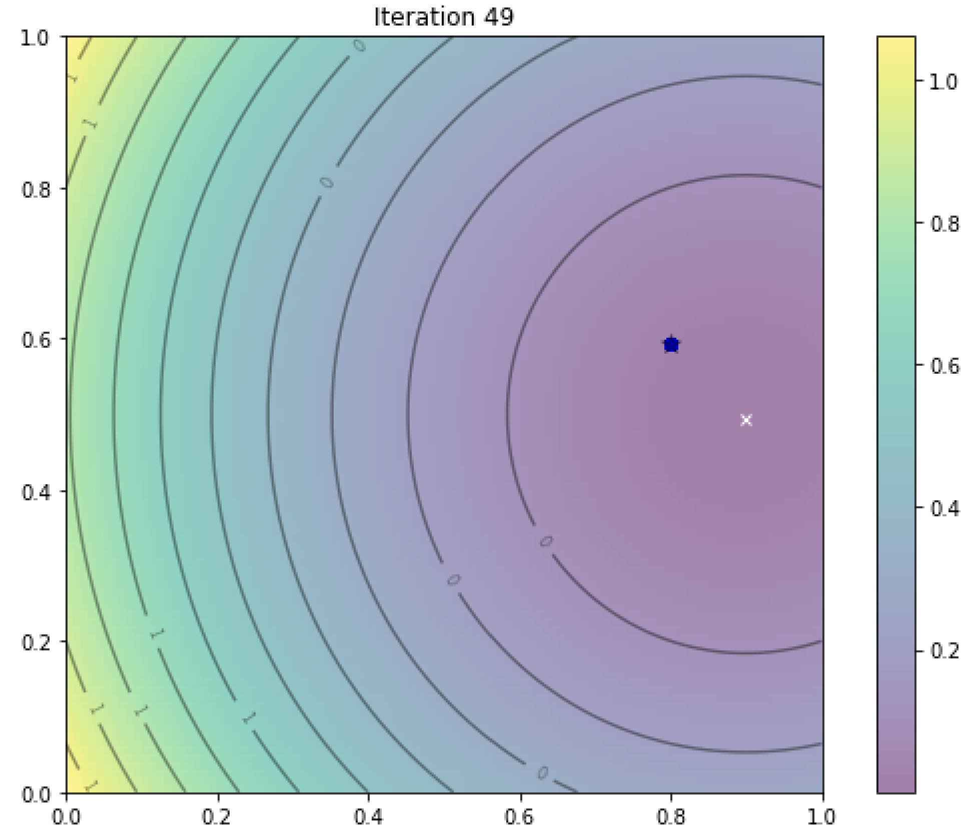
```

```

In [10]:

```


def f(x,y):
    "Objective function"
    return (x-0.9)**2 + (y-0.5)**2 # new

# Compute and plot the function in 3D within [0,5]x[0,5]
x, y = np.array(np.meshgrid(np.linspace(0,1,100), np.linspace(0,1,100))) # 1, nc
z = f(x, y)

# Find the global minimum
x_min = x.ravel()[z.argmin()]
y_min = y.ravel()[z.argmin()]

# Hyper-parameter of the algorithm
c1 = c2 = 0.1 # 0.1
w = 0.8 # 0.8

# Create particles
n_particles = 10 # 20
np.random.seed(1000) # take away or leave it here?
X = np.random.rand(2, n_particles)*0.1 + 0.2  # I can generate them randomly but
V = np.random.rand(2, n_particles)*0.1 + 0.2


# 0.2 + 0.2; 0.01 + 0.5

# with these parameters, we are already on the target:
# X = np.random.rand(2, n_particles)* 0.9
# V = np.random.rand(2, n_particles)*0.01
# also with 0.2, 0.4


#X = np.random.rand(2, n_particles) * 5
#V = np.random.randn(2, n_particles) * 0.1

# Initialize data
pbest = X
pbest_obj = f(X[0], X[1])
gbest = pbest[:, pbest_obj.argmin()]
gbest_obj = pbest_obj.min()

def update():
    "Function to do one iteration of particle swarm optimization"
    global V, X, pbest, pbest_obj, gbest, gbest_obj
    # Update params
    # r1, r2 = np.random.rand(2)
    r1, r2 = np.random.rand(2)
    V = w * V + c1*r1*(pbest - X) + c2*r2*(gbest.reshape(-1,1)-X)
    X = X + V
    obj = f(X[0], X[1])
    pbest[:, (pbest_obj >= obj)] = X[:, (pbest_obj >= obj)]
    pbest_obj = np.array([pbest_obj, obj]).min(axis=0)
    gbest = pbest[:, pbest_obj.argmin()]
    gbest_obj = pbest_obj.min()

# Set up base figure: The contour map
fig, ax = plt.subplots(figsize=(8,6))
fig.set_tight_layout(True)
```

```
img = ax.imshow(z, extent=[0, 1, 0, 1], origin='lower', cmap='viridis', alpha=0.
fig.colorbar(img, ax=ax)
ax.plot([x_min], [y_min], marker='x', markersize=5, color="white")
contours = ax.contour(x, y, z, 10, colors='black', alpha=0.4)
ax.clabel(contours, inline=True, fontsize=8, fmt="%.0f")
pbest_plot = ax.scatter(pbest[0], pbest[1], marker='o', color='black', alpha=0.5
p_plot = ax.scatter(X[0], X[1], marker='o', color='blue', alpha=0.5)
p_arrow = ax.quiver(X[0], X[1], V[0], V[1], color='blue', width=0.005, angles='x
gbest_plot = plt.scatter([gbest[0]], [gbest[1]], marker='*', s=100, color='black
ax.set_xlim([0,1])
ax.set_ylim([0,1])



def animate(i):
    "Steps of PSO: algorithm update and show in plot"
    title = 'Iteration {:02d}'.format(i)
    # Update params
    update()
    # Set picture
    ax.set_title(title)
    pbest_plot.set_offsets(pbest.T)
    p_plot.set_offsets(X.T)
    p_arrow.set_offsets(X.T)
    p_arrow.set_UVC(V[0], V[1])
    gbest_plot.set_offsets(gbest.reshape(1,-1))
    return ax, pbest_plot, p_plot, p_arrow, gbest_plot

anim = FuncAnimation(fig, animate, frames=list(range(1,50)), interval=500, blit=
anim.save("PSO.gif", dpi=120, writer="imagemagick")

print("PSO found best solution at f({})={}".format(gbest, gbest_obj))
print("Global optimal at f({})={}".format([x_min,y_min], f(x_min,y_min)))
```

```
2022-09-08 15:06:50,975 - matplotlib.animation - WARNING - MovieWriter
imagemagick unavailable; using Pillow instead.
2022-09-08 15:06:50,976 - matplotlib.animation - INFO - Animation.save
using <class 'matplotlib.animation.PillowWriter'>

PSO found best solution at f([0.79994233 0.59251027])=0.01856968734445
0914
Global optimal at f([0.8989898989898991, 0.494949494949495])=2.6527905
315783662e-05
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```