In [ ]:

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy
import math
```

In [ ]:

```python
def RBFkernel(x1: float, x2: float, sigma_sq: float = 1):
    """
    RBF kernel.
    """
    pow = -1/(2*sigma_sq) * (x1 - x2)**2
    return math.exp(pow)


def cov_mat(x1: np.ndarray, x2: np.ndarray, ker: callable, sigma_sq: float = 1) -> np.ndarray:
    """
    Returns the Covraiance matrix for the given kernel.
    """
    n = max(x1.shape)
    cov = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            cov[i][j] = ker(x1[i], x2[j], sigma_sq)

    return cov

def GPRegression(x1: np.ndarray, y1: np.ndarray, n_points: int,
                interval : tuple, sigma_sq: float) -> tuple:
    """
    GP Regression with specified parameters. Returns mean and
    standard deviation (for each point).
    """
    n = x1.__len__()

    x2 = np.linspace(interval[0], interval[1], n_points)#.reshape(-1, 1)
    cov = cov_mat(x1, x1, RBFkernel, sigma_sq=0.12)

    mean = np.zeros(n_points)
    stdev = np.zeros(n_points)
    i = 0
    while i < n_points:

        # First generate the mean, then stdev
        # Form k_x
        k_x = np.zeros(n)
        for j in range(n):
            k_x[j] = RBFkernel(x2[i], x1[j], 0.12)
        # Set kappa_x
        kappa_x = RBFkernel(x2[i], x2[i])
        # Make K + si_sq*I
        K_mod = cov + sigma_sq*np.eye(n)
        mean_right = scipy.linalg.solve(K_mod, y1, assume_a='pos')
        mean[i] = np.dot(k_x, mean_right)

        # Now do stdev
        stdev_right = scipy.linalg.solve(K_mod, k_x, assume_a='pos')
        if kappa_x - np.dot(k_x, stdev_right) < 0:
            print("Error, standard deviation too small. Increasing...")
            i = 0
            sigma_sq = sigma_sq*2
            continue

        stdev[i] = kappa_x - np.dot(k_x, stdev_right)
        stdev[i] = np.sqrt(stdev[i])
        i+=1
    print("Ending sigma: ", sigma_sq)
    return (mean, stdev)
```

In [ ]:

```python
# Fetching data
data = np.genfromtxt("gp.dat")
x1 = data[:, 0]
y1 = data[:, 1]

#Desired output
n_points = 200
x2 = np.linspace(0, 1, n_points)

interval = (0, 1)
sigma_sq = 1
# Get Mean, stdev
mean, stdev = GPRegression(x1, y1, n_points, interval, sigma_sq)
```

In [ ]:

```python
plt.scatter(x1, y1, s=10)
plt.plot(x2, mean, color='tab:red', label= "mean")
plt.plot(x2, mean+2*stdev, color='k', linestyle='--', label = "2$\sigma$")
plt.plot(x2, mean-2*stdev, color='k', linestyle='--')
plt.fill_between(x2, mean+2*stdev, mean, alpha=0.15, color='tab:pink')
plt.fill_between(x2, mean-2*stdev, mean, alpha=0.15, color='tab:pink')
```

```
plt.gca().set_facecolor((0.9, 0.9, 0.9))
plt.grid(True)
plt.title("Gaussian Process on gp.dat")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()
```

In [ ]:

```
n_draws = 20
K_mod = cov_mat(x2, x2, ker=RBFkernel, sigma_sq=0.12) + sigma_sq*np.eye(x2.__len__())
ys = np.random.multivariate_normal(mean=mean, cov=K_mod, size=n_draws)#.reshape(-1, 1)

for i in range(n_draws):
    plt.plot(x2, ys[i])
plt.gca().set_facecolor((0.9, 0.9, 0.9))
plt.grid(True)
plt.title(r"20 Draws from $\mathcal{G}\ (\mu', k')$")
plt.show()
```