In [ ]:
```python
import math
import numpy as np
import matplotlib.pyplot as plt
import sympy as sm
import scipy as scp
```

In [ ]:
```python
def gauleg(x1, x2, x, w, n):
    EPS = 3.0e-11
    m = (n + 1) // 2  # Find only half the roots because of symmetry
    xm = 0.5 * (x2 + x1)
    xl = 0.5 * (x2 - x1)
    for i in range(1, m + 1):
        z = math.cos(math.pi * (i - 0.25) / (n + 0.5))
        while True:
            p1 = 1.0
            p2 = 0.0
            for j in range(1, n + 1):
                # Recurrence relation
                p3 = p2
                p2 = p1
                p1 = ((2.0 * j - 1.0) * z * p2 - (j - 1.0) * p3) / j
            # Derivative
            pp = n * (z * p1 - p2) / (z * z - 1.0)
            z1 = z
            # Newton’s method
            z = z1 - p1 / pp
            if abs(z - z1) <= EPS:
                break
        x[i] = xm - xl * z
        x[n + 1 - i] = xm + xl * z
        # Weights
        w[i] = 2.0 * xl / ((1.0 - z * z) * pp * pp)
        w[n + 1 - i] = w[i]
```

In [ ]:
```python
def Gauss_Legendre_Quad(fs, weights: np.ndarray, zeros: np.ndarray) :
    sum = 0
    n = len(weights)
    y = sm.symbols('y')
    for i in range(n):
        sum += weights[i] * fs.subs(y, zeros[i])
    return sum

y = sm.symbols('y')
x = sm.symbols('x')
funcs = [1, x]
Tfuncs = [sm.exp(-(x - y)**2), x*sm.exp(-(x - y)**2)]
```

In [ ]:
```python
# Input the number of quadrature points
ns = [40, 80]
xspan = np.linspace(-1, 1, 100)
eigvals_40 = np.zeros((2, ns[0]))
eigvals_80 = np.zeros((2, ns[1]))
# Allocate arrays x and w

for n, N in enumerate(ns):
    xs = [0.0] * (N + 1)  # Zeros of Gauss-Legendre
    ws = [0.0] * (N + 1)

    # Call the gauleg function
    gauleg(-1.0, 1.0, xs, ws, N)

    for i, fs in enumerate(Tfuncs):
        res = Gauss_Legendre_Quad(fs, ws, xs)
        K = np.zeros((N, N))
        for k in range(N):
            for l in range(N):
                K[k][l] = np.exp(-(xs[l] - xs[k])**2) * ws[k]
        eigvals = scp.linalg.eigvals(K)

        if N == 40:
            for j in range(ns[0]):
                eigvals_40[i, j] = abs(eigvals[j])
        if N == 80:
            for j in range(ns[1]):
                eigvals_80[i, j] = abs(eigvals[j])
```

In [ ]:
```python
colors = ['tab:blue', 'tab:red']
eigs = [eigvals_40, eigvals_80]
fig, axs = plt.subplots(1, 2, figsize=(20, 10))
labels = ["f $\equiv$ 1", "f $\equiv$ x"]

for i in range(len(eigs[0])):
    # Plot the scatter plots and specify labels and markers
    axs[i].scatter(range(len(eigs[i][0])), eigs[i][0], c=colors[0], label=labels[0])
    axs[i].scatter(range(len(eigs[i][1])), eigs[i][1], c=colors[1], label=labels[1], marker='^')
    axs[i].set_yscale('log')
    axs[i].set_facecolor("#E6E6E6")
    axs[i].set_xlabel("Eigenvalues")
    axs[i].set_title(f"Spectra for $K = {ns[i]}$")
    axs[i].grid(True)
    axs[i].set_axisbelow(True)
    # Add legend for each plot
    axs[i].legend(loc='upper right', fontsize='large')

plt.tight_layout()
plt.show()
```