Task 3

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [82]:
```
def read_data():
  X = np.array(pd.read_csv('X.txt', header=None))
  Y = np.array(pd.read_csv('Y.txt', header=None))
  ones = np.ones((X.shape[0], 1))
  X = np.c_[ones, X]
  return X, Y
```

a) n=4

In [98]:
```
def task_a(X: np.ndarray, Y: np.ndarray, n: int=4) -> None:
  X = X[:n]
  Y = Y[:n]
  xtx = np.linalg.inv(np.matmul(X.T, X))
  ls = np.matmul(xtx, np.matmul(X.T, Y))
  return (xtx, ls.flatten())

X, Y = read_data()
xtx, betas = task_a(X, Y, n=4)
print(f'XTX^(-1): \n{xtx}')
print(f'Least-Squares for n=4:\nBeta 0: {betas[0]}\nBeta 1: {betas[1]}\nBeta 2:
```

```
XTX^(-1):
[[1.86905258e+09 1.73429460e+08 7.85725234e+07 3.79496696e+08]
 [1.73429460e+08 1.60950983e+07 7.28635452e+06 3.52145345e+07]
 [7.85725234e+07 7.28635452e+06 3.31066733e+06 1.59517851e+07]
 [3.79496696e+08 3.52145345e+07 1.59517851e+07 7.70542826e+07]]
Least-Squares for n=4:
Beta 0: 6294.731056213379
Beta 1: 612.530839920044
Beta 2: 218.07334327697754
Beta 3: 1287.5460586547852
```

b) Update step X^T X (n+1)

In [89]:
```
# Read the data
X, Y = read_data()

# Create a sub-matrix with n=4
X_4 = X[:4, ]
Y_4 = Y[:4]

# Calculate the new design matrix X^T X (n+1)
XTX = np.linalg.inv(X_4 @ X_4.T)
u = XTX @ X[4].T
v = 1 / (1 + X[4] @ u)
XTX = XTX - v * (u @ u.T)
XTX
```

Out[89]:
```
array([[-1.86739490e+09, -2.25935268e+09, -1.67065732e+09,
        -2.06428681e+09],
       [-2.25935268e+09, -1.08404739e+09, -2.84929596e+09,
        -1.66895707e+09],
       [-1.67065732e+09, -2.84929596e+09, -1.07902509e+09,
        -2.26273272e+09],
       [-2.06428681e+09, -1.66895707e+09, -2.26273272e+09,
        -1.86569571e+09]])
```

c) Sequential LS estimator for n > 4 using Sherman-Morrison formula

In [93]:
```
X, Y = read_data()

X_4 = X[:4]
Y_4 = Y[:4]

# Calculate A = (X^TX)^-1 for n = 4
A = np.linalg.inv(X_4.T @ X_4)

# Calculate the LS estimator for n = 201 using the batch formula
beta_201 = np.linalg.inv(X.T @ X) @ X.T @ Y

# List for ||beta_i - beta_201||
beta_diff = []

# Sequentially update (X^TX)^-1 and compute the LS estimator for n > 4
for i in range(5, X.shape[0] + 1):
    # Add a new data point to X_4 and the corresponding value to Y_4
    new_X = X[i-1:i, :]
    new_Y = Y[i-1:i]
    X_4 = np.vstack([X_4, new_X])
    Y_4 = np.append(Y_4, new_Y)

    # Sherman-Morrison formula
    u = A @ new_X.T
    v = 1 / (1 + new_X @ u)
    A = A - v * (u @ u.T)

    # LS estimator for specific n
    beta_i = A @ X_4.T @ Y_4

    # ||beta_n - beta_201||
    norm_diff = np.linalg.norm(beta_i - beta_201)

    beta_diff.append(norm_diff)
```

d) Plotting the LS difference

In [97]:
```
# Plotting
plt.plot(range(4, 201), beta_diff, label=r'$||\hat{\beta}_n - \hat{\beta}_{201}
plt.xlabel('n')
plt.ylabel(r'$||\hat{\beta}_n - \hat{\beta}_{201}||$')
plt.title('Task 3 (d)')
plt.legend()
plt.show()
```