Q7) Iterative Least Squares.

(a) as $k \to \infty$ (x^k) converges

ie $x^k = x^{k+1}$

$$A^T(Ax^k - b) = 0$$

so $A^TA x^k = A^Tb$

which is the normal equation

Normal equation has a unique solution $\hat{x}$

$x^k = \hat{x}$ when $k \to \infty$

(b) (i) For multiplication of A and x^k. $\Rightarrow O(2mn)$

(ii) For subtraction of Ax^k and b $\Rightarrow O(m)$

(iii) For multiplication with A^T $\Rightarrow O(mn)$

(iv) Norm calculation $\Rightarrow O(1)$ constant time (assumed)

(v) subtraction from x^k $\Rightarrow O(n)$

Total $= O(mn)$

$\Rightarrow$ Same thing for a total of k iterations

Total Complexity $= O(mnk)$

(c) Code attached below

⇒ Random generation of 30×10 matrix A and 30×1 random vector b.

⇒ Rank of matrix A = 10

⇒ We also plot the graph of error error = $\|x - \hat{x}\|$.

⇒ We can observe that as iterations increase error decreases, thus indicating and validating that iterative least squares converges on the least square solution.

(d) The usual method will be:-

(i) Compute the QR factorisation :- $A = QR$

(ii) Compute $Q^T b$

(iii) Solve $R\hat{x} = Q^T b$ by back substitution.

In the iterative method the only expensive operations are matrix to vector multiplication. So in case there are efficient and fast methods for these multiplications then the iterative method may become more computationally efficient compared to the normal method. In the normal method we are faced with the operation of QR factorisation which is expensive. In iterative method we can also change number of iterations if we want less accurate results for faster calculations.

Code used to implement the iterative least squares algorithm

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

def iterative_ls(A, b, num_iter=100):
    iter_array = []
    error_array = []
    (m, n) = A.shape
    x = np.zeros(shape=(n,))

    x_hat = np.linalg.lstsq(A, b, rcond=None)[0]

    for i in range(num_iter):
        x = x - (1 / np.linalg.norm(A, 2) ** 2) * np.dot(np.transpose(A), (np.dot(A, x) - b))
        iter_array.append(i + 1)
        error_array.append(np.linalg.norm(x - x_hat))

    print(f'\nNo. of iterations: {num_iter}')
    print('\nLeast squares solution computed using iterative method:\n', x)
    print('\nActual least squares solution:\n', x_hat)

    plt.plot(iter_array, error_array)
    plt.xlabel('No. of iterations')
    plt.ylabel('Error (Norm of (x_k - x_hat))')
    plt.show()

def main():
    m = 30
    n = 10
    A = np.random.rand(m, n)
    b = np.random.rand(m,)
    np.set_printoptions(linewidth=np.inf)
    print('\nMatrix A:\n', A)
    np.set_printoptions(linewidth=120)
    print('\nVector b:\n', b)
    np.set_printoptions(linewidth=np.inf)
    print(f'\nRank of A = {np.linalg.matrix_rank(A)}')

    iterative_ls(A, b)

if __name__ == '__main__':
    main()
```

Results on the terminal on running the code

```
suhas@suhas-HP-Pavilion-Laptop-15-cs1xxx:~$ /bin/python3 "/home/suhas/Semester-5/Linear Algebra/Assignment-2/Q7/iterative_ls

Matrix A:
 [[0.05214947 0.97514913 0.63864132 0.3221285  0.46419768 0.67615036 0.82423495 0.87667626 0.0248824  0.22184661]
 [0.22893271 0.84858988 0.70817235 0.27283201 0.74591965 0.12961437 0.08676714 0.14181797 0.79984795 0.14334843]
 [0.79981507 0.00856988 0.18333981 0.72500109 0.50381566 0.33284803 0.72306617 0.99146226 0.32007193 0.06263064]
 [0.76313068 0.86994067 0.36075659 0.54186918 0.44380051 0.30868768 0.75246761 0.53334808 0.49100306 0.24997721]
 [0.74839533 0.30164213 0.73430054 0.44278131 0.52754282 0.57263592 0.60257912 0.77336505 0.23705228 0.40656949]
 [0.96436847 0.9037281  0.10229293 0.67979034 0.32020704 0.66026172 0.60541713 0.0333685  0.65439582 0.03230528]
 [0.10223347 0.91502363 0.80433775 0.37528966 0.53076926 0.61021135 0.87490345 0.41173247 0.57527153 0.27985689]
 [0.72893311 0.61078496 0.13506557 0.15603241 0.05176519 0.68956946 0.24226284 0.1142703  0.78782555 0.44417102]
 [0.29510119 0.01922151 0.0054421  0.75834505 0.5259067  0.01470225 0.58882308 0.37228508 0.60985033 0.21518776]
 [0.20749086 0.3596283  0.6233907  0.70630402 0.308612   0.22207728 0.49907859 0.37484685 0.97509062 0.37473791]
 [0.22223993 0.57257354 0.81956592 0.08394284 0.09466368 0.87404501 0.11424406 0.28392614 0.16599088 0.62271905]
 [0.4895405  0.55371643 0.76798647 0.47567832 0.35889613 0.7948751  0.33522175 0.95298513 0.9923682  0.32986099]
 [0.95932717 0.01345754 0.6332868  0.79657995 0.61205994 0.49646102 0.57120351 0.22783743 0.44697937 0.87835774]
 [0.13190173 0.82457693 0.35918527 0.56959042 0.77369248 0.25525854 0.99013833 0.32582761 0.48337432 0.62923194]
 [0.6771195  0.82822287 0.12271848 0.13855585 0.70355947 0.05706518 0.07610499 0.75829949 0.05639393 0.43213823]
 [0.15093875 0.33608318 0.86933227 0.72788892 0.11052494 0.80469055 0.9984962  0.66671383 0.06503576 0.25137762]
 [0.20587289 0.85674495 0.99132861 0.77738766 0.42303059 0.42233428 0.73632782 0.34903808 0.86221291 0.51397693]
 [0.38697619 0.4573479  0.34839025 0.41617084 0.87670359 0.08301856 0.79091134 0.41058393 0.39941907 0.1600049 ]
 [0.05426744 0.86680505 0.19102438 0.21146902 0.39712194 0.38076408 0.07141323 0.44928081 0.29022602 0.48099452]
 [0.67647443 0.71406551 0.69835457 0.88548865 0.60737597 0.72402625 0.35733347 0.55754531 0.82455038 0.00251725]
 [0.36661436 0.72691603 0.47946928 0.05342914 0.87073411 0.04678392 0.36514428 0.10219136 0.23886164 0.77928146]
 [0.42974953 0.48510813 0.57358244 0.17836094 0.78018877 0.5355223  0.40818083 0.56541057 0.01463928 0.5212078 ]
 [0.58033116 0.08525234 0.87913332 0.34238144 0.89166181 0.55670612 0.32964879 0.23467901 0.38610226 0.13480559]
 [0.80275403 0.66375697 0.13127106 0.50357221 0.17253283 0.33795499 0.57150938 0.51526435 0.73475793 0.69327926]
 [0.64452139 0.35707853 0.7296544  0.25978625 0.8607617  0.65645615 0.64271275 0.35556489 0.8481878  0.13902963]
 [0.95054736 0.87600223 0.37025534 0.11552999 0.59658417 0.23434788 0.03273558 0.19600638 0.2750702  0.20761952]
 [0.50722835 0.89547493 0.05077762 0.69840002 0.14875266 0.08072415 0.89697692 0.63165029 0.26292192 0.7111419 ]
 [0.06755431 0.77005253 0.95859789 0.40946684 0.00414498 0.85030685 0.9618499  0.72772627 0.34263523 0.59754883]
 [0.80766687 0.73784704 0.19395671 0.45204581 0.62153367 0.26378176 0.32717268 0.84406679 0.82886952 0.34289101]
 [0.80287136 0.13498186 0.94339413 0.56601142 0.08204833 0.90505109 0.00522843 0.98220833 0.591316   0.63387238]]

Vector b:
 [0.95615439 0.12520596 0.88295464 0.58957177 0.82251506 0.64916857 0.28828143 0.84401914 0.01226722 0.98346574
 0.77408176 0.91301307 0.72224735 0.35615211 0.24394065 0.35819836 0.53166009 0.20375825 0.11749003 0.84295461
 0.71102062 0.19312365 0.93409358 0.94172913 0.7749397  0.56326992 0.12372992 0.21684187 0.39742011 0.74288888]

Rank of A = 10

No. of iterations: 100

Least squares solution computed using iterative method:
 [ 0.41598713 -0.07910126  0.17983166 -0.05648514  0.0462705   0.34939218  0.02263253  0.03297432  0.21729621  0.04979413]

Actual least squares solution:
 [ 0.45506486 -0.10919391  0.17859361 -0.19919864  0.02480554  0.3500679   0.11317073  0.03924352  0.26922958  0.04644452]
```

Graph of error obtained v/s number of iterations on running the least square algorithm