DATE
PAGE

AI61003 Linear Algebra for AI & ML
Assignment 02 - Problem 07

Consider left invertible matrix $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $x^{(i)} \in \mathbb{R}^n$ for $i = 1, 2, 3 \ldots$. $x^{(1)} = 0$ and for $k = 1, 2, 3 \ldots$

$$x^{(k+1)} = x^{(k)} - \frac{1}{\|A\|^2} A^T \left(A x^{(k)} - b\right)$$

$$\hat{x} = (A^T A)^{-1} A^T b$$

(a) Let $C = I - \frac{A^T A}{\|A\|^2}$, $d = \frac{A^T b}{\|A\|^2}$

$\therefore\ x^{(k+1)} = C x^{(k)} + d$ for $k = 1, 2 \ldots$

$$\begin{aligned}
x^{(k+1)} &= C x^{(k)} + d \\
&= C\left(C x^{(k-1)} + d\right) + d \\
&= C^2 x^{(k-1)} + (C + I) d \\
&= C^2 \left(C x^{(k-2)} + d\right) + (C + I) d \\
&= C^3 x^{(k-2)} + (C^2 + C + I) d \\
&\ \vdots \\
x^{(k+1)} &= C^k x^{(1)} + (C^{k-1} + C^{k-2} + \ldots + C + I) d
\end{aligned}$$

(Put $x^{(1)} = 0$)

$$x^{(k+1)} = D_k d$$

where $D_k = I + \sum_{i=1}^{k-1} C^i$

Let $\lim_{k \to \infty} D_k = D$

$D = I + C + C^2 + C^3 \ldots\ldots$ — (1)
$CD = C + C^2 + C^3 + C^4 \ldots\ldots$ — (2)

(1) − (2) $\Rightarrow (I - C)D = I$

$I - C = I - \left(I - \frac{A^TA}{\|A\|^2}\right) = \frac{A^TA}{\|A\|^2}$

($I-C$ is invertible $\because$ gram-matrix A^TA is invertible $\because$ A has linearly independent columns).

$\therefore D = (I-C)^{-1} = \|A\|^2 (A^TA)^{-1}$

$\lim_{k \to \infty} x^{(k+1)} = \lim_{k\to\infty} D_k d = \left(\lim_{k\to\infty} D\right) d$
$= \|A\|^2 (A^TA)^{-1} \frac{A^Tb}{\|A\|^2}$

$\Rightarrow \lim_{k\to\infty} x^{(k+1)} = (A^TA)^{-1} A^Tb = \hat{x}$

Hence sequence $\{x^{(k)}\}$ converges to $\hat{x}$ as k tends to ∞.

(b) Let $ct(y)$ be the computat^n cost for entity (y).
$ct(x^{(1)}) = n$ because all n-entries need to be initialized to 0.
$ct(x^{(k+1)}) = ct(x^{(k)}) + ct(Cx^{(k)})$
$+ ct(Cx^{(k)} + d)$
for $k \geq 1$.

DATE
PAGE

$$ct(x^{(k+1)}) = ct(x^{(k)}) + n^2 + n$$

$$\begin{aligned} ct(x^{(k+1)}) &= ct(x^{(k-1)}) + 2(n^2+n) \\ &= ct(x^{(1)}) + k(n^2+n) \\ &= n + kn(n+1) + \lambda \end{aligned}$$

Here $\lambda = ct(c) + ct(d)$

Note that $c = I - \dfrac{A^T A}{\|A\|^2}$

$$d = A^T b / \|A\|^2$$

$$\begin{aligned} \lambda = {} & ct(\|A\|^2) + ct(A^T) + ct(A^T A) \\ & + ct(A^T A / \|A\|^2) + ct(I - A^T A/\|A\|^2) \\ & + ct(A^T b) + ct(A^T b / \|A\|^2) \end{aligned}$$

$$\begin{aligned} \lambda = {} & (2mn-1) + (mn) + (n^2 m) \\ & + n^2 + n^2 + (mn) + n \end{aligned}$$

$$\lambda = n^2(m+2) + 4mn + n - 1$$

$$\therefore \quad ct(x^{(k)}) = (k-1)n(n+1) + n^2(m+2) + 4mn + 2n - 1$$

for $k > 1$

and $ct(x^{(1)}) = n$

(Note that c and d have to be computed just once & $\therefore$ were not shown in the recursive formula)

(c) Solution/code attached after part (d).

(d) Suppose the iterative method takes k iterat^{n}s to converge to $\hat{x}$ within some acceptable range of precision. So the cost cst_1 will be $O(n^2(m+k))$ (from the $cst(x^{(k)})$ formula derived in part (b)).

$\therefore\ cst_1 = O(n^2(m+k))$

To compute direct sol^{n} of $A^TA\hat{x} = A^Tb$ we do QR decomposition using $(n-1)$ reflectors.

In rth step, cst to compute Q_r will be $O((m-r+1)^2)$ and the cost of multiplicat^{n}s on both sides would be $m^2(n+1)$.

$$cst_2 = \sum_{r=1}^{n-1} O((m-r+1)^2) + m^2(n+1) + \lambda$$

$$= O(m^3 - (m-n)^3) + m^2(n^2-1) + \lambda$$

$$= O(nm^2) + m^2(n^2-1) + O(m)$$

($\lambda = O(m)$ is the cost of solving back-substitution)

$\therefore\ cst_2 = O(m^2n^2)$.

Given that $m \geq n$, iterative method will be better when m is very large (which usually is the case when we are fitting a model to some 'm' observat^{n}s). Also for smaller m, if $k = O(m^2)$ then also iterative method

can be used.
Since k depends on the precision bound we choose, there has to be a trade-off between computation cost and precision. So if we always choose k bounded (upper) by m^2, then iterative method is computationally more beneficial.

AI61003 Linear Algebra for AI & ML

Assignment 02 - Problem 07(c)

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

In [2]:
```python
A = np.random.random((30,10))
while ( np.linalg.matrix_rank(A) != 10 ) :
    A = np.random.random((30,10))
A
```

Out[2]:
```
array([[0.81980106, 0.42996646, 0.41615019, 0.98430451, 0.93769169,
        0.58924779, 0.14459998, 0.44589338, 0.61616546, 0.18664659],
       [0.29107368, 0.06805186, 0.36074467, 0.42482159, 0.3545573 ,
        0.3450397 , 0.5074016 , 0.5150124 , 0.3092849 , 0.89517556],
       [0.32151458, 0.79746882, 0.954158  , 0.27269424, 0.02404356,
        0.64509807, 0.56109553, 0.31085449, 0.81704991, 0.78111818],
       [0.11971831, 0.88079203, 0.11966085, 0.13687322, 0.61148393,
        0.30866281, 0.81607906, 0.73583471, 0.24833083, 0.73040046],
       [0.4815946 , 0.10399506, 0.65647304, 0.11455872, 0.91596897,
        0.81292872, 0.53984366, 0.1149081 , 0.04396539, 0.33583807],
       [0.85383771, 0.68203454, 0.67873883, 0.6558534 , 0.83623116,
        0.55723611, 0.91480051, 0.96374728, 0.77412123, 0.0260353 ],
       [0.31426056, 0.98280783, 0.53492562, 0.60532414, 0.21599211,
        0.42912918, 0.21266872, 0.12017493, 0.66048191, 0.52448224],
       [0.54810691, 0.13374302, 0.07774595, 0.60410785, 0.38799723,
        0.16074371, 0.71045098, 0.34372977, 0.53973293, 0.50902906],
       [0.38366538, 0.62302586, 0.89389346, 0.95025641, 0.85836699,
        0.20360161, 0.74591915, 0.91590952, 0.83114474, 0.23014518],
       [0.92438232, 0.29977718, 0.59180457, 0.06342131, 0.44287027,
        0.46416829, 0.43064176, 0.96906091, 0.27234166, 0.2845511 ],
       [0.20462559, 0.30465399, 0.48182444, 0.71067542, 0.70876339,
        0.33477579, 0.34423615, 0.93459177, 0.10093228, 0.83065679],
       [0.63500087, 0.68794475, 0.15508567, 0.01135181, 0.84850045,
        0.82840067, 0.64978488, 0.19204162, 0.03204361, 0.54380025],
       [0.16282619, 0.79200229, 0.48821751, 0.7833585 , 0.69572059,
        0.58541616, 0.31012815, 0.56515481, 0.42800159, 0.50353083],
       [0.82273882, 0.7021514 , 0.62276854, 0.83429301, 0.04599685,
        0.64578558, 0.59926309, 0.39768276, 0.42934445, 0.20895766],
       [0.96061186, 0.77539804, 0.40683559, 0.52687573, 0.7415859 ,
        0.74423827, 0.99049875, 0.84828223, 0.26360248, 0.38218057],
       [0.98067119, 0.23628664, 0.09800299, 0.34001359, 0.06018575,
        0.07995535, 0.7142485 , 0.23706771, 0.519438  , 0.71931112],
       [0.49041783, 0.42244442, 0.16046367, 0.69652082, 0.7219762 ,
        0.65634297, 0.84270132, 0.45833186, 0.61286096, 0.12233731],
       [0.17816601, 0.7896144 , 0.33801624, 0.82760683, 0.78153822,
        0.40534441, 0.79888435, 0.60277682, 0.45901291, 0.2521117 ],
       [0.16904507, 0.13581976, 0.24510051, 0.1616989 , 0.68286133,
        0.48122541, 0.32136895, 0.91705931, 0.59659212, 0.31909352],
       [0.88368057, 0.8690898 , 0.0544669 , 0.05906441, 0.98058663,
        0.08838753, 0.73928041, 0.10764699, 0.79046888, 0.8146803 ],
       [0.33481186, 0.66816816, 0.35443772, 0.73958502, 0.85332037,
        0.05165348, 0.487779  , 0.3755133 , 0.74786882, 0.99614905],
       [0.15174328, 0.66172835, 0.63719409, 0.87020065, 0.40137977,
        0.06665967, 0.70127195, 0.28845003, 0.95781089, 0.39756189],
       [0.81236705, 0.8514267 , 0.91991452, 0.97458269, 0.5182986 ,
        0.48650803, 0.02244829, 0.50451316, 0.93027415, 0.49094939],
       [0.11981365, 0.86366056, 0.83811764, 0.48499536, 0.06086256,
        0.45154169, 0.32418712, 0.25549257, 0.70359518, 0.04899737],
       [0.55785317, 0.72283637, 0.39401701, 0.15357451, 0.54493467,
        0.58412555, 0.91456881, 0.20005191, 0.98797197, 0.57049667],
       [0.38170023, 0.60182646, 0.19231264, 0.65547251, 0.65976714,
        0.55764093, 0.21238237, 0.46266941, 0.81495384, 0.4449997 ],
       [0.69831849, 0.31797701, 0.02222312, 0.68901288, 0.04860829,
        0.09842817, 0.312874  , 0.46986162, 0.74021504, 0.21319171],
       [0.12234586, 0.90497426, 0.93205661, 0.97755675, 0.57417258,
        0.956913  , 0.53943996, 0.19228697, 0.85047273, 0.84012811],
       [0.47635546, 0.11045758, 0.44875562, 0.65269748, 0.49220525,
        0.97447606, 0.20535875, 0.15956735, 0.13524868, 0.6890902 ],
       [0.73113694, 0.06219508, 0.23476929, 0.69011119, 0.6169071 ,
        0.85889576, 0.53954772, 0.39676034, 0.82613368, 0.41090069]])
```

In [3]:
```python
b = np.random.random((30,1))
b
```

Out[3]:
```
array([[0.9191341 ],
       [0.40263561],
       [0.05076489],
       [0.17102826],
       [0.57805781],
       [0.11518178],
       [0.98061788],
       [0.40685732],
       [0.0150959 ],
       [0.67988549],
       [0.17869277],
       [0.25771637],
       [0.40772774],
       [0.8273833 ],
       [0.01153663],
       [0.76857179],
       [0.41669874],
       [0.44037906],
       [0.03080953],
       [0.05987474],
       [0.05280064],
       [0.47283542],
       [0.13334114],
       [0.65478657],
       [0.85649523],
       [0.79392571],
       [0.80332512],
       [0.27556753],
       [0.4526037 ],
       [0.02776457]])
```

In [4]:
```python
def Iterative_Least_Squares_Solution ( A , b , iters = 100 ) :
    m, n = A.shape # A is an mxn matrix
    x = np.zeros((n, 1)) # x(0) is an n-vector with all zeros
    x_ks = [ x ] # x_ks stores the intermediate solns from all iterations
                 # x_ks = [x(0), x(1), ..., x(100)]

    for it in range(iters) :
        # implement formula x(k+1) = x(k) - A_t . (Ax(k) - b) / ||A||^2
        # where A_t is transpose of A and ||.|| is Frobenius norm
        t = np.matmul(A, x) - b
        t = np.matmul(np.transpose(A), t)
        t = t / (np.linalg.norm(A) ** 2)
        x = x - t
        x_ks.append(x)

    return x, x_ks

def Least_Squares_Solution ( A , b ) :
    # Compute and return (A_t . A)_inv . A_t . b
    # where A_t is transpose of A and B_inv is inverse of B
    t = np.matmul(np.transpose(A), A)
    t = np.linalg.inv(t)
    t = np.matmul(t, np.transpose(A))
    x_h = np.matmul(t, b)
    return x_h # x_h stands for x(hat). It is the exact least-squares solution.
```

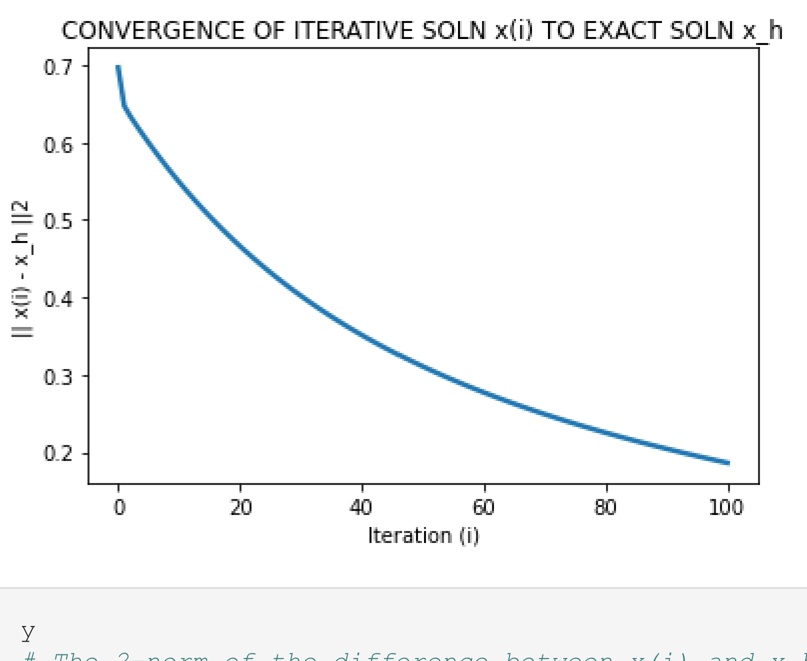
In [5]:
```python
iterative_soln, iterations = Iterative_Least_Squares_Solution(A, b, iters = 100)
iterative_soln # iterative LS solution after 100 iterations
```

Out[5]:
```
array([[ 0.34539319],
       [ 0.13326228],
       [-0.07038971],
       [ 0.23618725],
       [-0.213896  ],
       [ 0.23340391],
       [ 0.001701  ],
       [-0.11116346],
       [ 0.12455191],
       [ 0.03540093]])
```

In [6]:
```python
direct_soln = Least_Squares_Solution(A, b)
direct_soln # exact LS solution using direct formula
```

Out[6]:
```
array([[ 0.35861194],
       [ 0.19713635],
       [-0.19178886],
       [ 0.28446213],
       [-0.29176198],
       [ 0.30905134],
       [-0.01797044],
       [-0.08369256],
       [ 0.10310435],
       [ 0.0419859 ]])
```

In [7]:
```python
# In order to prove the convergence of iteratve solution to the exact solution x_h,
# we plot the 2-norm of the difference between x(i) and x_h as i progresses from 0
# to 100. Here, ||.||2 stands for 2-norm.
y = [np.linalg.norm(t - direct_soln) for t in iterations]
step = list(range(0, 101))
plt.plot(step, y, linewidth = 2.5)
plt.xlabel('Iteration (i)')
plt.ylabel('|| x(i) - x_h ||2')
plt.title('CONVERGENCE OF ITERATIVE SOLN x(i) TO EXACT SOLN x_h')
plt.show()
# Clearly, the plot is monotonically decreasing and converges towards 0. Hence proved
# that the algorithm converges to the exact solution x_h.
```



In [8]:
```python
y
# The 2-norm of the difference between x(i) and x_h decreases from 0.697 to 0.186
# over 100 iterations. Also, if you compare, the entries of "iterative_soln" vector
# with "direct_soln" they turn out to be pretty close.
```

Out[8]:
```
[0.6967965429432338,
 0.6473404141850903,
 0.6337938174605499,
 0.6220973528992194,
 0.6107881147625894,
 0.5997953157712982,
 0.5891061302227172,
 0.5787104773089428,
 0.5685987260127596,
 0.5587615868688481,
 0.5491900947074819,
 0.5398755957837982,
 0.5308097356811119,
 0.5219844478053426,
 0.5133919424310147,
 0.5050246962669844,
 0.4968754425120481,
 0.4889371613722774,
 0.48120307101354065,
 0.473666618924196,
 0.46632147366440124,
 0.4591615169798816,
 0.4521808362593117,
 0.44537371731573233,
 0.43873463747361235,
 0.4322582589443068,
 0.42593942247372985,
 0.4197731412470888,
 0.41375459503648304,
 0.40787912457808745,
 0.4021422261665002,
 0.39653954645464407,
 0.3910668774483805,
 0.3857201516857081,
 0.38049543759110066,
 0.37538893499616754,
 0.3703969708184177,
 0.36551599489046377,
 0.36074257593251885,
 0.3560733976615288,
 0.3515052550307292,
 0.3470350505938383,
 0.34265979098848726,
 0.3383765835338497,
 0.33418263293776773,
 0.3300752381089826,
 0.3260517890703603,
 0.3221097639692708,
 0.3182467261815185,
 0.3144603215054457,
 0.310748275443037,
 0.3071083905650382,
 0.30353854395727603,
 0.3000366847455253,
 0.2966008316964081,
 0.2932290708919482,
 0.2899195534755182,
 0.28667049346702944,
 0.2834801656453155,
 0.2803469034957469,
 0.277269097221205,
 0.2742451918146109,
 0.2712736851912823,
 0.26835312637944925,
 0.2654821137673211,
 0.2626592934051492,
 0.2598833573607795,
 0.25715304212723467,
 0.2544671270809077,
 0.25182443298898727,
 0.24922382056477255,
 0.2466641890695687,
 0.24414447495988684,
 0.24166365057870218,
 0.2392207228895536,
 0.23681473225229394,
 0.23444475123932923,
 0.23210988349120718,
 0.22980926261044254,
 0.22754205109248907,
 0.22530743929279204,
 0.2231046444288764,
 0.2209329096164499,
 0.21879150293852032,
 0.21667971654654855,
 0.21459686579268025,
 0.21254228839211886,
 0.21051534361472518,
 0.20851541150494693,
 0.20654189212920426,
 0.20459420484987498,
 0.2026717876250457,
 0.20077409633321225,
 0.19890060412213476,
 0.19705080078107012,
 0.19522419213562522,
 0.19342029946449277,
 0.19163865893735058,
 0.1898788210732237,
 0.18814035021862743,
 0.18642282404482696]
```