Linear Algebra
Problem Set 4 - Solutions

Abolfazl Ranjbar

Course Instructors: Dr. Elham Ghasrodashti and Dr. Peyman Adibi

Computer Engineering Department
University of Isfahan

Fall Semester 2024

11.2 *Left and right inverses of a vector.* Suppose that x is a nonzero n-vector with $n > 1$.

(a) Does x have a left inverse?

(b) Does x have a right inverse?

$x = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$

In each case, if the answer is yes, give a left or right inverse; if the answer is no, give a specific nonzero vector and show that it is not left- or right-invertible.

(a) yes : The pesudo-inverse $\longrightarrow x^+ = (x^T x)^{-1} x^T = (1/\|x\|^2) x^T$

(b) no : The matrix rows must be linearly independent, but x is a tall matrix. $(n \times 1)$

11.5 *Inverse of a block matrix.* Consider the $(n+1) \times (n+1)$ matrix

$$A = \begin{bmatrix} I & a \\ a^T & 0 \end{bmatrix}, \rightarrow \begin{bmatrix} I_{n\times n} & [1]^n \\ \underbrace{[\quad\quad]}_{n} & 0 \end{bmatrix} = A_{(n+1)\times(n+1)}$$

where a is an n-vector.

(a) When is A invertible? Give your answer in terms of a. Justify your answer.

↳ when $a \neq 0$

A columns must be linearly independent to be invertable

↳ $Ax = 0 \rightarrow x = 0$

$$\begin{bmatrix} I & a \\ a^T & 0 \end{bmatrix} \begin{bmatrix} \gamma \\ \beta \end{bmatrix} = 0 \rightarrow \begin{matrix} \gamma + \beta a = 0 \\ \gamma a^T = 0 \end{matrix} \begin{cases} \text{if } a \neq 0 \rightarrow \gamma = 0 \\ \beta a = 0 \rightarrow \beta = 0 \end{cases}$$

else: $a = 0 \rightarrow \gamma, \beta$ can get none zero values.

(b) Assuming the condition you found in part (a) holds, give an expression for the inverse matrix A^{-1}.

book solution

$$A^{-1} = \frac{1}{\|a\|^2} \begin{bmatrix} \|a\|^2 I - aa^T & a \\ a^T & -1 \end{bmatrix} \rightarrow A^{-1}Ax = A^{-1}b$$

$= x_2 a \rightarrow$ scalar

$$\begin{bmatrix} I & a \\ a^T & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \begin{cases} x_1 + a x_2 = b_1 \\ a^T x_1 = b_2 \end{cases}$$

$$b_2 = a^T(b_1 - x_2 a) = a^T b_1 - x_2 \|a\|^2 \rightarrow x_2 = (a^T b_1 - b_2)/\|a\|^2$$

$$\rightarrow x_1 = b_1 - a((a^T b_1 - b_2)/\|a\|^2) \rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \frac{1}{\|a\|^2} \begin{bmatrix} \|a\|^2 I - aa^T & a \\ a^T & -1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$A^{-1}Ax = A^{-1}b$

$(s_k)_i = 0$ for $i > k$

$Rs_k = e_k \rightarrow s_k = R^{-1} e_k$

11.7 *Inverse of an upper triangular matrix.* Suppose the $n \times n$ matrix R is upper triangular and invertible, *i.e.*, its diagonal entries are all nonzero. Show that R^{-1} is also upper triangular. *Hint.* Use back substitution to solve $Rs_k = e_k$, for $k = 1, \ldots, n$, and argue that $(s_k)_i = 0$ for $i > k$.

$s_k = R^{-1} e_k \rightarrow (s_k)_i = 0 \rightarrow$ for $i > k \rightarrow R^{-1}$ upper triangular

$(s_k)_n = b_n / R_{nn} = 0 \rightarrow x_i = 0 \quad (s_k)_k = 1/R_{kk} \rightarrow R^{-1}$

$k < n \rightarrow (s_k)_n = \dfrac{b_n}{R_{nn}} \rightarrow x_i = 0 \rightarrow (s_k)_k = \dfrac{1}{R_{kk}} \rightarrow R^{-1}$ upper triangular

11.16 *Inverse of running sum matrix.* Find the inverse of the $n \times n$ running sum matrix,

$$S = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

$S^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -1 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix}$

Does your answer make sense?

$Sx = I$

$x_{1j} = 0$

$x_{1j} + x_{2j} = 0$

$\vdots$

$x_{1j} + x_{2j} + \cdots + x_{j-1,j} = 0$

$x_{1j} + x_{2j} + \cdots + x_{jj} = 1$

$x_{1j} + x_{2j} + \cdots + x_{j+1,j} = 0$

$\vdots$

$x_{1j} + x_{2j} + \cdots + x_{nj} = 0$

if $j < n$:

$x_{1j} = x_{j-1,j} = 0$

$x_{jj} = 1, \quad x_{j+1,j} = -1$

$x_{j+2,j} = \cdots = x_{nj} = 0$

if $j = n$:

$x_{1n} = \cdots = x_{j-1,j} = 0$

$x_{nn} = 1$

11.17 *A matrix identity.* Suppose A is a square matrix that satisfies $A^k = 0$ for some integer k. (Such a matrix is called *nilpotent*.) A student guesses that $(I-A)^{-1} = I+A+\cdots+A^{k-1}$, based on the infinite series $1/(1-a) = 1+a+a^2+\cdots$, which holds for numbers a that satisfy $|a| < 1$.

Is the student right or wrong? If right, show that her assertion holds with no further assumptions about A. If she is wrong, give a counterexample, *i.e.*, a matrix A that satisfies $A^k = 0$, but $I + A + \cdots + A^{k-1}$ is not the inverse of $I - A$.

$(I-A)(I+A+\cdots+A^{k-1}) = I+A+\cdots+A^{k-1} - (A+A^2+\cdots+A^k)$

$\rightsquigarrow = I + A^k = I \longrightarrow$ student is right. $\rightsquigarrow$ $I-A$ is invertible

11.22 *Properties of pseudo-inverses.* For an $m \times n$ matrix A and its pseudo-inverse $A^\dagger$, show that $A = AA^\dagger A$ and $A^\dagger = A^\dagger AA^\dagger$ in each of the following cases.

(a) A is tall with linearly independent columns.

$A^\dagger A = I \rightsquigarrow \underbrace{AA^\dagger A}_{I} = A$, $\underbrace{A^\dagger AA^\dagger}_{I} = A^\dagger$

(b) A is wide with linearly independent rows.

$AA^\dagger = I \rightsquigarrow \underbrace{AA^\dagger A}_{I} = A$, $A^\dagger \underbrace{AA^\dagger}_{I} = A^\dagger$

(c) A is square and invertible.

$A^\dagger = A^{-1} \rightsquigarrow \underbrace{AA^\dagger A}_{I} = A$, $\underbrace{A^\dagger AA^\dagger}_{I} = A^\dagger$

11.23 *Product of pseudo-inverses.* Suppose A and D are right-invertible matrices and the product AD exists. We have seen that if B is a right inverse of A and E is a right inverse of D, then EB is a right inverse of AD. Now suppose B is the pseudo-inverse of A and E is the pseudo-inverse of D. Is EB the pseudo-inverse of AD? Prove that this is always true or give an example for which it is false.

example:

$A = \begin{bmatrix}1 & 1\end{bmatrix}$, $D = \begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}$, $AD = \begin{bmatrix}1 & 2\end{bmatrix}$

$A^\dagger = \frac{1}{2}\begin{bmatrix}1\\ 1\end{bmatrix}$, $D^\dagger = \begin{bmatrix}1 & 0\\ 0 & 1/2\end{bmatrix}$, $(AD)^\dagger = \frac{1}{5}\begin{bmatrix}1\\ 2\end{bmatrix}$

but $\rightsquigarrow D^\dagger A^\dagger = \begin{bmatrix}1/2\\ 1/4\end{bmatrix} \neq (AD)^\dagger$ $\rightsquigarrow$ so that result above is false.

12.1 *Approximating a vector as a multiple of another one.* In the special case $n = 1$, the general least squares problem (12.1) reduces to finding a scalar x that minimizes $\|ax - b\|^2$, where a and b are m-vectors. (We write the matrix A here in lower case, since it is an m-vector.) Assuming a and b are nonzero, show that $\|a\hat{x} - b\|^2 = \|b\|^2(\sin\theta)^2$, where $\theta = \angle(a, b)$. This shows that the optimal relative error in approximating one vector by a multiple of another one depends on their angle.

$$\hat{x} = (a^Ta)^{-1}a^Tb = \frac{a^Tb}{\|a\|^2}$$

$$\|a\hat{x} - b\|^2 = \left\|a\frac{a^Tb}{\|a\|^2} - b\right\|^2 = \|a\|^2\frac{(a^Tb)^2}{\|a\|^4} - 2\frac{(a^Tb)^2}{\|a\|} + \|b\|^2$$

$$= \|b\|^2\left(1 - \frac{a^Tb}{\|a\|^2\|b\|^2}\right) = \|b\|^2(1 - (\cos\theta)^2) = \|b\|^2(\sin\theta)^2$$

12.3 *Least angle property of least squares.* Suppose the $m \times n$ matrix A has linearly independent columns, and b is an m-vector. Let $\hat{x} = A^\dagger b$ denote the least squares approximate solution of $Ax = b$.

(a) Show that for any n-vector x, $(Ax)^Tb = (Ax)^T(A\hat{x})$, *i.e.*, the inner product of Ax and b is the same as the inner product of Ax and $A\hat{x}$. *Hint.* Use $(Ax)^Tb = x^T(A^Tb)$ and $(A^TA)\hat{x} = A^Tb$.

$$(Ax)^Tb = x^T(A^TA\hat{x}) = (Ax)^T(A\hat{x})$$

(b) Show that when $A\hat{x}$ and b are both nonzero, we have

$$\frac{(A\hat{x})^Tb}{\|A\hat{x}\|\|b\|} = \frac{\|A\hat{x}\|}{\|b\|}.$$

The left-hand side is the cosine of the angle between $A\hat{x}$ and b. *Hint.* Apply part (a) with $x = \hat{x}$.

$$(A\hat{x})^Tb = \|A\hat{x}\|^2 \xrightarrow{\text{divide}} \frac{}{\|A\hat{x}\|\|b\|} \text{ to get the equality}$$

(c) *Least angle property of least squares.* The choice $x = \hat{x}$ minimizes the distance between Ax and b. Show that $x = \hat{x}$ also minimizes the angle between Ax and b. (You can assume that Ax and b are nonzero.) *Remark.* For any positive scalar α, $x = \alpha\hat{x}$ also minimizes the angle between Ax and b.

$$\cos^{-1}\left(\frac{(A\hat{x})^Tb}{\|A\hat{x}\|\|b\|}\right) \leq \cos^{-1}\left(\frac{(Ax)^Tb}{\|Ax\|\|b\|}\right)$$

$$\rightsquigarrow \|A\hat{x}\|\|Ax\| \geq (Ax)^T(A\hat{x})$$

$$\rightsquigarrow \text{by Cauchy-Schwarz inequality}$$

12.4 *Weighted least squares.* In least squares, the objective (to be minimized) is

$$\|Ax - b\|^2 = \sum_{i=1}^{m} (\tilde{a}_i^T x - b_i)^2,$$

where $\tilde{a}_i^T$ are the rows of A, and the n-vector x is to chosen. In the *weighted least squares problem*, we minimize the objective

$$\sum_{i=1}^{m} w_i (\tilde{a}_i^T x - b_i)^2,$$

where w_i are given positive weights. The weights allow us to assign different weights to the different components of the residual vector. (The objective of the weighted least squares problem is the square of the weighted norm, $\|Ax - b\|_w^2$, as defined in exercise 3.28.)

(a) Show that the weighted least squares objective can be expressed as $\|D(Ax - b)\|^2$ for an appropriate diagonal matrix D. This allows us to solve the weighted least squares problem as a standard least squares problem, by minimizing $\|Bx - d\|^2$, where $B = DA$ and $d = Db$.

$$\sum_{i=1}^{m} w_i (\tilde{a}_i^T x - b_i)^2 = \sum_{i=1}^{m} (\sqrt{w_i}(\tilde{a}_i^T x - b_i))^2 = \|D(Ax-b)\|^2$$

$$\rightarrow D = \begin{bmatrix} \sqrt{w_1} & 0 & \cdots & 0 \\ 0 & \sqrt{w_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & \sqrt{w_m} \end{bmatrix}$$

(b) Show that when A has linearly independent columns, so does the matrix B.

$Bx = DAx = 0 \rightarrow$ since D is invertible and by assumption A has linearly independent columns so $Ax = 0$ and $x = 0$ $\rightarrow$ then B is linearly independent

(c) The least squares approximate solution is given by $\hat{x} = (A^T A)^{-1} A^T b$. Give a similar formula for the solution of the weighted least squares problem. You might want to use the matrix $W = \mathbf{diag}(w)$ in your formula.

$$(B^T B)^{-1} B^T d = ((DA)^T (DA))^{-1} (DA)^T (Db)$$
$$= (A^T D^2 A)^{-1} (A^T D^2 b) \rightarrow W = D^2$$
$$= (A^T W A)^{-1} (A^T W b)$$

12.5 *Approximate right inverse.* Suppose the tall $m \times n$ matrix A has linearly independent columns. It does not have a right inverse, *i.e.*, there is no $n \times m$ matrix X for which $AX = I$. So instead we seek the $n \times m$ matrix X for which the residual matrix $R = AX - I$ has the smallest possible matrix norm. We call this matrix the *least squares approximate right inverse* of A. Show that the least squares right inverse of A is given by $X = A^\dagger$. *Hint.* This is a matrix least squares problem; see page 233.

$$\|R\|^2 = \sum_{j=1}^{m} \|r_j\|^2 = \sum_{j=1}^{m} \|Ax_j - e_j\|^2$$

$\hat{x}_j = A^\dagger e_j \longrightarrow$ using x_j to minimize $\|Ax_j - e_j\|^2$

$$\hat{X} = [\hat{x}_1 \ \hat{x}_2 \ \cdots \ \hat{x}_m] = [A^\dagger e_1 \ A^\dagger e_2 \ \cdots \ A^\dagger e_m]$$
$$= A^\dagger [e_1 \ e_2 \ \cdots \ e_m] = A^\dagger I = A^\dagger$$

12.8 *Least squares and QR factorization.* Suppose A is an $m \times n$ matrix with linearly independent columns and QR factorization $A = QR$, and b is an m-vector. The vector $A\hat{x}$ is the linear combination of the columns of A that is closest to the vector b, *i.e.*, it is the projection of b onto the set of linear combinations of the columns of A.

(a) Show that $A\hat{x} = QQ^T b$. (The matrix QQ^T is called the *projection matrix*.)

$$A\hat{x} = AA^\dagger b = QRR^{-1}Q^T b = QQ^T b$$

(b) Show that $\|A\hat{x} - b\|^2 = \|b\|^2 - \|Q^T b\|^2$. (This is the square of the distance between b and the closest linear combination of the columns of A.)

$A^T A\hat{x} = A^T b$

$$\|A\hat{x} - b\|^2 = \|b\|^2 + \|A\hat{x}\|^2 - \hat{x}^T A^T b$$

$\hat{x}^T A^T A\hat{x} \longrightarrow 2\|A\hat{x}\|^2$

$$= \|b\|^2 - \|A\hat{x}\|^2$$

$$\|A\hat{x} - b\|^2 = \|b\|^2 - \|QQ^T b\|^2 = \|b\|^2 - \|Q^T b\|^2$$

$$\|Qy\|^2 = y^T Q^T Q y = y^T y = \|y\|^2$$

12.9 *Invertibility of matrix in sparse least squares formulation.* Show that the $(m+n) \times (m+n)$ coefficient matrix appearing in equation (12.11) is invertible if and only if the columns of A are linearly independent.

$$\begin{bmatrix} 0 & A^T \\ A & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & A^T \\ A & I \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \longrightarrow$$ if columns of A are linearly dependent $\rightarrow x \neq 0$

$\hookrightarrow$ matrix is not invertible

$A^T y = 0$, $Ax + y = 0 \rightsquigarrow y = -Ax$

$\hookrightarrow -A^T Ax = 0 \longrightarrow x^T A^T Ax = \|Ax\|^2 = 0 \longrightarrow Ax = 0 \rightsquigarrow x = 0$

$\hookrightarrow$ linearly independent

$\hookrightarrow$ matrix is invertible

13.15 *Estimating a matrix.* Suppose that the n-vector x and the m-vector y are thought to be approximately related by a linear function, *i.e.*, $y \approx Ax$, where A is an $m \times n$ matrix. We do not know the matrix A, but we do have observed data,

$$x^{(1)}, \ldots, x^{(N)}, \qquad y^{(1)}, \ldots, y^{(N)}.$$

We can estimate or guess the matrix A by choosing it to minimize

$$\sum_{i=1}^{N} \|Ax^{(i)} - y^{(i)}\|^2 = \|AX - Y\|^2,$$

where $X = [x^{(1)} \cdots x^{(N)}]$ and $Y = [y^{(1)} \cdots y^{(N)}]$. We denote this *least squares estimate* as $\hat{A}$. (The notation here can be confusing, since X and Y are known, and A is to be found; it is more conventional to have symbols near the beginning of the alphabet, like A, denote known quantities, and symbols near the end, like X and Y, denote variables or unknowns.)

(a) Show that $\hat{A} = YX^\dagger$, assuming the rows of X are linearly independent. *Hint.* Use $\|AX - Y\|^2 = \|X^T A^T - Y^T\|^2$, which turns the problem into a matrix least squares problem; see page 233.

$W = A^T \rightarrow \|X^T W - Y^T\| = \sum_{k=1}^{m} \|X^T w_k - y_k\|^2$

$\rightsquigarrow \hat{w}_k = (XX^T)^{-1} X y_k$; $\hat{A} = \hat{W}^T = [\hat{w}_1 \; \hat{w}_2 \cdots \hat{w}_k]^T$

$= [(XX^T)^{-1} X y_1 \;\; (XX^T)^{-1} X y_2 \cdots (XX^T)^{-1} X y_k]^T$

$= ((XX^T)^{-1} X Y^T)^T = YX^T (XX^T)^{-1}$

$= YX^\dagger$

(b) Suggest a good way to compute $\hat{A}$, and give the complexity in terms of n, m, and N.

by QR factorization $X^T = QR \rightarrow X^{-1} = X^T (XX^T)^{-1}$

$= QR (R^T R)^{-1} = QRR^{-1} R^{-T} = QR^{-T}$

$\hat{A} = YQR^{-T}$ computation steps :

- computing QR from an $N \times n$ matrix $X^T \rightarrow 2Nn^2$ flops
- Computing the product YQ ($m \times N$ - matrix and $N \times n$ - matrix) $\rightarrow 2mnN$ flops
- computing $m \times n$ matrix $\underbrace{YQ}_{m \times n} R^{-T}$ by $R^T \hat{A}^T = Z^T$ (forward substitution) $\rightarrow mn^2$ flops

$\hookrightarrow$ complexity : $2Nn^2 + 2mnN + mn^2$

15.3 *Weighted Gram matrix.* Consider a multi-objective least squares problems with matrices $A_1, \ldots, A_k$ and positive weights $\lambda_1, \ldots, \lambda_k$. The matrix

$$G = \lambda_1 A_1^T A_1 + \cdots + \lambda_k A_k^T A_k$$

is called the *weighted Gram matrix*; it is the Gram matrix of the stacked matrix $\tilde{A}$ (given in (15.2)) associated with the multi-objective problem. Show that G is invertible provided there is no nonzero vector x that satisfies $A_1 x = 0, \ldots, A_k x = 0$.

$Gx = 0$

$0 = x^T G x = \lambda_1 x^T A_1^T A_1 x + \cdots + \lambda_k x^T A_k^T A_k x$

$= \lambda_1 \|A_1 x\|^2 + \cdots + \lambda_k \|A_k x\|^2.$

$A_1 x = 0, \ldots, A_k x = 0 \rightarrow x = 0 \rightarrow G$ is invertible

15.4 *Robust approximate solution of linear equations.* We wish to solve the square set of n linear equations $Ax = b$ for the n-vector x. If A is invertible the solution is $x = A^{-1}b$. In this exercise we address an issue that comes up frequently: We don't know A exactly. One simple method is to just choose a typical value of A and use it. Another method, which we explore here, takes into account the variation in the matrix A. We find a set of K versions of A, and denote them as $A^{(1)}, \ldots, A^{(K)}$. (These could be found by measuring the matrix A at different times, for example.) Then we choose x so as to minimize

$$\|A^{(1)}x - b\|^2 + \cdots + \|A^{(K)}x - b\|^2,$$

the sum of the squares of residuals obtained with the K versions of A. This choice of x, which we denote x^{rob}, is called a *robust* (approximate) solution. Give a formula for x^{rob}, in terms of $A^{(1)}, \ldots, A^{(K)}$ and b. (You can assume that a matrix you construct has linearly independent columns.) Verify that for $K = 1$ your formula reduces to $x^{\text{rob}} = (A^{(1)})^{-1}b$.

$$\min \left(\left\| \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(k)} \end{bmatrix} x - \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(k)} \end{bmatrix} \right\| \right) \rightarrow \hat{A} = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(k)} \end{bmatrix}, \quad \hat{b} = \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(k)} \end{bmatrix}$$

$$\rightarrow \hat{x} = (\hat{A}^T \hat{A})^{-1} \hat{A}^T \hat{b} = \left(\sum_{k=1}^{K} (A^{(k)})^T A^{(k)} \right)^{-1} \left(\sum_{k=1}^{K} (A^{(k)})^T b^{(k)} \right)$$

15.9 *Regularizing stratified models.* In a *stratified* model (see page 272), we divide the data into different sets, depending on the value of some (often Boolean) feature, and then fit a separate model for each of these two data sets, using the remaining features. As an example, to develop a model of some health outcome we might build a separate model for women and for men. In some cases better models are obtained when we encourage the different models in a stratified model to be close to each other. For the case of stratifying on one Boolean feature, this is done by choosing the two model parameters $\theta^{(1)}$ and $\theta^{(2)}$ to minimize

$$\|A^{(1)}\theta^{(1)} - y^{(1)}\|^2 + \|A^{(2)}\theta^{(2)} - y^{(2)}\|^2 + \lambda\|\theta^{(1)} - \theta^{(2)}\|^2,$$

where $\lambda \geq 0$ is a parameter. The first term is the least squares residual for the first model on the first data set (say, women); the second term is the least squares residual for the second model on the second data set (say, men); the third term is a regularization term that encourages the two model parameters to be close to each other. Note that when $\lambda = 0$, we simply fit each model separately; when λ is very large, we are basically fitting one model to all the data. Of course the choice of an appropriate value of λ is obtained using out-of-sample validation (or cross-validation).

(a) Give a formula for the optimal $(\hat{\theta}^{(1)}, \hat{\theta}^{(2)})$. (If your formula requires one or more matrices to have linearly independent columns, say so.)

$\min(\|A\theta - b\|^2)$ where $A = \begin{bmatrix} A^{(1)} & 0 \\ 0 & A^{(2)} \\ \sqrt{\lambda} I & -\sqrt{\lambda} I \end{bmatrix}$, $\theta = \begin{bmatrix} \theta^{(1)} \\ \theta^{(2)} \end{bmatrix}$, $b = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ 0 \end{bmatrix}$

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

$= \begin{bmatrix} (A^{(1)})^T A^{(1)} + \lambda I & -\lambda I \\ -\lambda I & (A^{(2)})^T A^{(2)} + \lambda I \end{bmatrix} \begin{bmatrix} (A^{(1)})^T y^{(1)} \\ (A^{(2)})^T y^{(2)} \end{bmatrix}$

if A has linearly independent columns:

$A = \begin{bmatrix} A^{(1)} \\ A^{(2)} \end{bmatrix}$ ↳ stacked matrix

$Ax = 0 \rightsquigarrow Ax = \begin{bmatrix} A^{(1)} & 0 \\ 0 & A^{(2)} \\ \sqrt{\lambda} I & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} A^{(1)} x_1 \\ A^{(2)} x_2 \\ \sqrt{\lambda}(x_1 - x_2) \end{bmatrix} = 0 \begin{cases} A^{(1)} x_1 = 0 \\ A^{(2)} x_2 = 0 \end{cases}$

↳ $x_1 = x_2$ $\rightsquigarrow$ $x_1, x_2 = 0$

(b) *Stratifying across age groups.* Suppose we fit a model with each data point representing a person, and we stratify over the person's *age group*, which is a range of consecutive ages such as 18–24, 24–32, 33–45, and so on. Our goal is to fit a model for each age of k groups, with the parameters for adjacent age groups similar, or not too far, from each other. Suggest a method for doing this.

- we need to minimize : $\|A^{(1)}\theta^{(1)} - y^{(1)}\|^2 + \cdots + \|A^{(k)}\theta^{(k)} - y^{(k)}\|^2$

$+ \lambda\|\theta^{(1)} - \theta^{(2)}\|^2 + \lambda\|\theta^{(2)} - \theta^{(3)}\|^2 + \cdots + \lambda\|\theta^{(k-1)} - \theta^{(k)}\|^2$

15.10 *Estimating a periodic time series.* (See §15.3.2.) Suppose that the T-vector y is a measured time series, and we wish to approximate it with a P-periodic T-vector. For simplicity, we assume that $T = KP$, where K is an integer. Let $\hat{y}$ be the simple least squares fit, with no regularization, *i.e.*, the P-periodic vector that minimizes $\|\hat{y} - y\|^2$. Show that for $i = 1, \ldots, P-1$, we have

$$\hat{y}_i = \frac{1}{K} \sum_{k=1}^{K} y_{i+(k-1)P}.$$

In other words, each entry of the periodic estimate is the average of the entries of the original vector over the corresponding indices.

consisting of K copies of $I_{k \times K}$ stacked on top of each other

$A^T A = KI$, $A^T y = \sum_{k=1}^{K} y((k-1)P+1):(kP)$

$\hat{y} = (A^T A)^{-1} A^T y = \frac{1}{K} \sum_{k=1}^{K} y((k-1)P+1):(kP)$

$\hat{y}_i = \frac{1}{K} \sum_{k=1}^{K} y_{(k-1)P+i}$

15.11 *General pseudo-inverse.* In chapter 11 we encountered the pseudo-inverse of a tall matrix with linearly independent columns, a wide matrix with linearly independent rows, and a square invertible matrix. In this exercise we describe the pseudo-inverse of a general matrix, *i.e.*, one that does not fit these categories. The general pseudo-inverse can be defined in terms of Tikhonov regularized inversion (see page 317). Let A be any matrix, and $\lambda > 0$. The Tikhonov regularized approximate solution of $Ax = b$, *i.e.*, unique minimizer of $\|Ax - b\|^2 + \lambda\|x\|^2$, is given by $(A^TA + \lambda I)^{-1}A^Tb$. The pseudo-inverse of A is defined as

$$A^\dagger = \lim_{\lambda \to 0} (A^TA + \lambda I)^{-1}A^T.$$

In other words, $A^\dagger b$ is the limit of the Tikhonov-regularized approximate solution of $Ax = b$, as the regularization parameter converges to zero. (It can be shown that this limit always exists.) Using the kernel trick identity (15.10), we can also express the pseudo-inverse as

$$A^\dagger = \lim_{\lambda \to 0} A^T(AA^T + \lambda I)^{-1}.$$

(a) What is the pseudo-inverse of the $m \times n$ zero matrix?

$0^\dagger = 0 \rightsquigarrow 0^\dagger = \lim_{\lambda \to 0} (0^T0 + \lambda I)^{-1}0^T = \lim_{\lambda \to 0} 0^T = 0^T$

(b) Suppose A has linearly independent columns. Explain why the limits above reduce to our previous definition, $A^\dagger = (A^TA)^{-1}A^T$.

A^TA is invertible

$\hookrightarrow \lim_{\lambda \to 0} (A^TA + \lambda I)^{-1} = (A^TA)^{-1} \rightsquigarrow$ so $A^\dagger = (A^TA)^{-1}A^T$

(c) Suppose A has linearly independent rows. Explain why the limits above reduce to our previous definition, $A^\dagger = A^T(AA^T)^{-1}$.

Hint. For parts (b) and (c), you can use the fact that the matrix inverse is a continuous function, which means that the limit of the inverse of a matrix is the inverse of the limit, provided the limit matrix is invertible.

$A^\dagger = \lim_{\lambda \to 0} (AA^T + \lambda I)^{-1} = (AA^T)^{-1}$

16.1 *Smallest right inverse.* Suppose the $m \times n$ matrix A is wide, with linearly independent rows. Its pseudo-inverse $A^\dagger$ is a right inverse of A. In fact, there are many right inverses of A and it turns out that $A^\dagger$ is the smallest one among them, as measured by the matrix norm. In other words, if X satisfies $AX = I$, then $\|X\| \geq \|A^\dagger\|$. You will show this in this problem.

(a) Suppose $AX = I$, and let $x_1, \ldots, x_m$ denote the columns of X. Let b_j denote the jth column of $A^\dagger$. Explain why $\|x_j\|^2 \geq \|b_j\|^2$. *Hint.* Show that $z = b_j$ is the vector of smallest norm that satisfies $Az = e_j$, for $j = 1, \ldots, m$.

(b) Use the inequalities from part (a) to establish $\|X\| \geq \|A^\dagger\|$.

(a) z that minimize $\|z\|^2$

$\hookrightarrow Az = e_j \rightsquigarrow A^\dagger e_j = b_j$

(b) X is right inverse of $A \rightsquigarrow Ax_j = e_j$, we know that $\|x_j\|^2 \geq \|b_j\|^2$

$\hookrightarrow \|X\|^2 = \sum_{j=1}^{m} \|x_j\|^2 \geq \sum_{j=1}^{m} \|b_j\|^2 = \|A^\dagger\|^2$

16.2 *Matrix least norm problem.* The matrix least norm problem is

$$\begin{array}{ll} \text{minimize} & \|X\|^2 \\ \text{subject to} & CX = D, \end{array}$$

where the variable to be chosen is the $n \times k$ matrix X; the $p \times n$ matrix C and the $p \times k$ matrix D are given. Show that the solution of this problem is $\hat{X} = C^\dagger D$, assuming the rows of C are linearly independent. *Hint.* Show that we can find the columns of X independently, by solving a least norm problem for each one.

$\|X\|^2 = \sum_{j=1}^{k} \|x_j\|^2 \longrightarrow \hat{x} = [C^\dagger d_1 \;\; C^\dagger d_2 \;\; \cdots \;\; C^\dagger d_k] = C^\dagger D$

16.3 *Closest solution to a given point.* Suppose the wide matrix A has linearly independent rows. Find an expression for the point x that is closest to a given vector y (*i.e.*, minimizes $\|x - y\|^2$) among all vectors that satisfy $Ax = b$.

Remark. This problem comes up when x is some set of inputs to be found, $Ax = b$ represents some set of requirements, and y is some nominal value of the inputs. For example, when the inputs represent actions that are re-calculated each day (say, because b changes every day), y might be yesterday's action, and the today's action x found as above gives the least change from yesterday's action, subject to meeting today's requirements.

suppose $z = x - y$. $\longrightarrow$ $A(x-y) = b - Ay \rightarrow Az = b - Ay$

so $z = A^\dagger(b - Ay) \longrightarrow x = z + y = A^\dagger(b - Ay) + y$

16.4 *Nearest vector with a given average.* Let a be an n-vector and β a scalar. How would you find the n-vector x that is closest to a among all n-vectors that have average value β? Give a formula for x and describe it in English.

minimizing $\|x - a\|^2$ subject to $1^T x = n\beta$

$$\begin{bmatrix} 2I & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2a \\ n\beta \end{bmatrix} \quad \begin{cases} 2x + 1y = 2a \rightarrow x = a - (y/2)1 \\ 1^T x = n\beta \end{cases}$$

$1^T a - ny/2 = n\beta \longrightarrow y = 2(\beta - 1^T a/n) = 2(\beta - \text{avg}(a))$

$\rightsquigarrow x = a - (y/2)1 = a + (\beta - \text{avg}(a))1$

16.6 *Modifying a diet to meet nutrient requirements.* (Continuation of exercise 8.9.) The current daily diet is specified by the n-vector d^{curr}. Explain how to find the closest diet d^{mod} to d^{curr} that satisfies the nutrient requirements given by the m-vector n^{des}, and has the same cost as the current diet d^{curr}.

$\min(\|d - d^{\text{curr}}\|^2)$ subject to $Nd = n^{\text{des}}$, $c^T d = c^T d^{\text{curr}}$

$$\rightsquigarrow \begin{bmatrix} N \\ c^T \end{bmatrix} x = \begin{bmatrix} n^{\text{des}} - Nd^{\text{curr}} \\ 0 \end{bmatrix} \rightarrow \hat{x} = \begin{pmatrix} \begin{bmatrix} N \\ c^T \end{bmatrix}^T \begin{bmatrix} n^{\text{des}} - Nd^{\text{curr}} \\ 0 \end{bmatrix}$$

$x = d - d^{\text{curr}}$

$$\hat{x} = [N^T \; c] \begin{pmatrix} \begin{bmatrix} NN^T & Nc \\ c^T N^T & c^T c \end{bmatrix} \begin{bmatrix} n^{\text{des}} - Nd^{\text{curr}} \\ 0 \end{bmatrix}$$

16.9 *Smoothest force sequence to move a mass.* We consider the same setup as the example given on page 343, where the 10-vector f represents a sequence of forces applied to a unit mass over 10 1-second intervals. As in the example, we wish to find a force sequence f that achieves zero final velocity and final position one. In the example on page 343, we choose the smallest f, as measured by its norm (squared). Here, though, we want the *smoothest* force sequence, *i.e.*, the one that minimizes

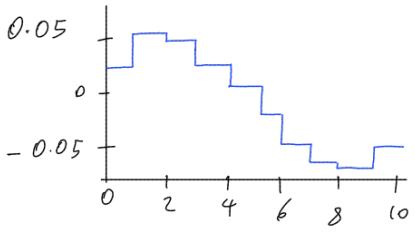
$$f_1^2 + (f_2 - f_1)^2 + \cdots + (f_{10} - f_9)^2 + f_{10}^2.$$

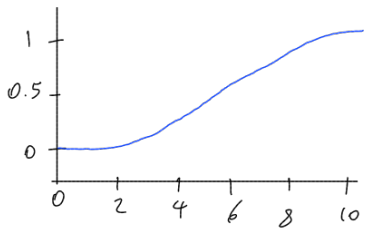
(This is the sum of the squares of the differences, assuming that $f_0 = 0$ and $f_{11} = 0$.) Explain how to find this force sequence. Plot it, and give a brief comparison with the force sequence found in the example on page 343.

$Cf = d$ where $\begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ 19/2 & 17/2 & \cdots & 3/2 & 1/2 \end{bmatrix}$, $d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

we need to min $f_1^2 + (f_2 - f_1)^2 + \cdots + (f_{10} - f_9)^2 + f_{10}^2 = \|Af\|^2$

$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$ { minimize $\|Af\|^2$, subject $Cf = d$ $\rightsquigarrow$ $f = (0.035, 0.049, 0.0466, 0.0326, 0.0117, -0.0117, -0.0326, -0.0466, -0.049, 0.035)$





16.11 *Least distance problem.* A variation on the least norm problem (16.2) is the least distance problem,

$$\begin{array}{ll} \text{minimize} & \|x - a\|^2 \\ \text{subject to} & Cx = d, \end{array}$$

where the n-vector x is to be determined, the n-vector a is given, the $p \times n$ matrix C is given, and the p-vector d is given. Show that the solution of this problem is

$$\hat{x} = a - C^\dagger(Ca - d),$$

assuming the rows of C are linearly independent. *Hint.* You can argue directly from the KKT equations for the least distance problem, or solve for the variable $y = x - a$ instead of x.

$\begin{bmatrix} 2I & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} 2a \\ d \end{bmatrix}$ { $2x + C^T z = 2a$, $Cx = d$

$\rightsquigarrow x = a - (1/2) C^T z \longrightarrow 1/2\, CC^T z = Ca - d$

$\rightsquigarrow z = 2(CC^T)^{-1}(Ca - d) \longrightarrow x = a - C^T(CC^T)^{-1}(Ca - d)$

$= a - C^\dagger(Ca - d)$

$\rightsquigarrow y = x - a \rightarrow \min(\|y\|^2)$ subject $Cy = d - Ca \rightsquigarrow \hat{x} = a + \hat{y} = a - C^\dagger(Ca - d)$

16.14 *Invertibility of matrix in sparse constrained least squares formulation.* Show that the $(m+n+p) \times (m+n+p)$ coefficient matrix appearing in equation (16.11) is invertible if and only if the KKT matrix is invertible, *i.e.*, the conditions (16.5) hold.

A has linearly independent columns:

$$\begin{bmatrix} 0 & A^T \\ A & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \begin{array}{l} A^T y = 0, \quad Ax + y = 0 \rightarrow y = -Ax \\ A^T Ax = 0 \rightarrow x^T A^T Ax = \|Ax\|^2 = 0 \end{array}$$

$\rightarrow$ so this matrix is invertible $\leftarrow$

but if $Ax = 0, x \neq 0$:

$$\begin{bmatrix} 0 & A^T \\ A & I \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \text{since } (x, 0) \neq 0 \rightarrow \text{is not invertible}$$

QR Decomposition of Orthogonal, Scaled, and Broken Matrices

This script demonstrates the QR decomposition process on various types of matrices: an orthogonal matrix, a scaled version of the orthogonal matrix, and a modified (broken) version of the scaled matrix. It performs the decomposition using NumPy's np.linalg.qr function and examines the resulting R matrix, as well as verifies the orthogonality of Q by checking if $Q^T \cdot Q$ approximates the identity matrix. In each step, values close to zero in the R and $Q^T \cdot Q$ matrices are set to zero for numerical stability. The code highlights how QR decomposition behaves when the matrix is scaled or broken by altering its elements.

```
import numpy as np
threshold = 1e-10
np.random.seed(42)
matrix = np.random.randn(6, 6)
Q, _ = np.linalg.qr(matrix)
orthogonal_matrix = Q
Q, R = np.linalg.qr(orthogonal_matrix)
R[np.abs(R) < threshold] = 0
Q_T_Q = np.dot(Q.T, Q)
Q_T_Q[np.abs(Q_T_Q) < threshold] = 0
print("Step 1: Decomposition of orthogonal matrix")
print("R:")
print(R)
print("Q^T Q:")
print(Q_T_Q)
print("-" * 50)
norms = np.arange(10, 16)
modified_matrix = orthogonal_matrix * norms
Q, R = np.linalg.qr(modified_matrix)
R[np.abs(R) < threshold] = 0
Q_T_Q = np.dot(Q.T, Q)
Q_T_Q[np.abs(Q_T_Q) < threshold] = 0
print("Step 2: Decomposition of scaled matrix")
print("R:")
print(R)
print("Diagonal values of R:", np.diag(R))
print("Q^T Q:")
print(Q_T_Q)
print("-" * 50)
broken_matrix = modified_matrix.copy()
broken_matrix[0, 3] = 0
Q, R = np.linalg.qr(broken_matrix)
R[np.abs(R) < threshold] = 0
Q_T_Q = np.dot(Q.T, Q)
Q_T_Q[np.abs(Q_T_Q) < threshold] = 0
print("Step 3: Decomposition of broken matrix")
print("R:")
```

```python
print(R)
print("Q^T Q:")
print(Q_T_Q)
print("-" * 50)
```

```
Step 1: Decomposition of orthogonal matrix
R:
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]]
Q^T Q:
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]]
--------------------------------------------------
Step 2: Decomposition of scaled matrix
R:
[[10.  0.  0.  0.  0.  0.]
 [ 0. 11.  0.  0.  0.  0.]
 [ 0.  0. 12.  0.  0.  0.]
 [ 0.  0.  0. 13.  0.  0.]
 [ 0.  0.  0.  0. 14.  0.]
 [ 0.  0.  0.  0.  0. 15.]]
Diagonal values of R: [10. 11. 12. 13. 14. 15.]
Q^T Q:
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]]
--------------------------------------------------
Step 3: Decomposition of broken matrix
R:
[[10.          0.          0.          1.96745568  0.          0.        ]
 [ 0.         11.          0.         -0.60873227  0.          0.        ]
 [ 0.          0.         12.         -2.73422903  0.          0.        ]
 [ 0.          0.          0.          9.49212949 -1.80930822 -8.24964565]
 [ 0.          0.          0.          0.         13.88259355 -1.07517026]
 [ 0.          0.          0.          0.          0.         12.4814805 ]]
Q^T Q:
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]]
```

Visualization of QR Decomposition and Matrix Differences

This script visualizes the process of QR decomposition on a randomly generated 6x6 matrix A. It uses Matplotlib to display the original matrix A, the orthogonal matrix Q, and the upper triangular matrix R resulting from the decomposition. Additionally, the difference between the original matrix A and the product QR is shown to illustrate how well the decomposition reconstructs the matrix. The plot also displays the matrix product $Q^T \cdot Q$ to confirm the orthogonality of Q. The images are rendered using grayscale color mapping with specified limits to highlight the matrix elements.

```
import matplotlib.pyplot as plt
from matplotlib import gridspec
# create a random matrix
A = np.random.randn(6,6)
# QR decomposition
Q,R = np.linalg.qr(A)
# show the matrices
fig = plt.figure(figsize=(10,6))
axs = [0]*5
c = 1.5 # color limits
gs1 = gridspec.GridSpec(2,6)
axs[0] = plt.subplot(gs1[0,:2])
axs[0].imshow(A,vmin=-c,vmax=c,cmap='gray')
axs[0].set_title('A',fontweight='bold')
axs[1] = plt.subplot(gs1[0,2:4])
axs[1].imshow(Q,vmin=-c,vmax=c,cmap='gray')
axs[1].set_title('Q',fontweight='bold')
axs[2] = plt.subplot(gs1[0,4:6])
axs[2].imshow(R,vmin=-c,vmax=c,cmap='gray')
axs[2].set_title('R',fontweight='bold')
axs[3] = plt.subplot(gs1[1,1:3])
axs[3].imshow(A - Q@R,vmin=-c,vmax=c,cmap='gray')
axs[3].set_title('A - QR',fontweight='bold')
axs[4] = plt.subplot(gs1[1,3:5])
axs[4].imshow(Q.T@Q,cmap='gray')
axs[4].set_title(r'Q$^T$Q',fontweight='bold')
# remove ticks from all axes
for a in axs:
  a.set_xticks([])
  a.set_yticks([])
plt.tight_layout()
plt.savefig('Figure_09_01.png',dpi=300)
plt.show()
```

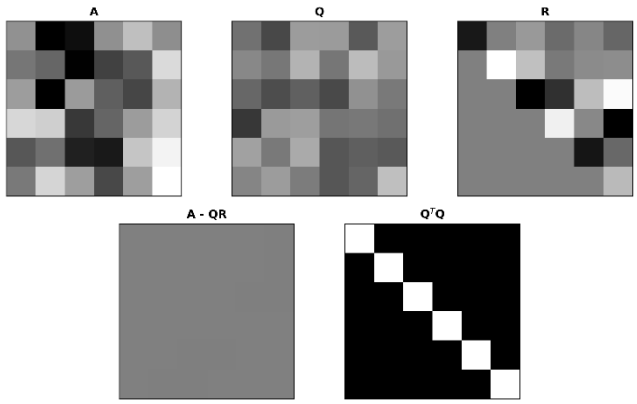


Figure 1-Output

Linear Regression for Happiness vs. Number of Courses Taken

This script explores the relationship between the number of courses taken and general life happiness using linear regression. Initially, a scatter plot is created to visualize the data points. The script then builds a statistical model by calculating the regression coefficients using the left-inverse method of the design matrix. Once the model is fitted, predicted happiness values are calculated and plotted alongside the real data, with residuals (errors) shown as dashed lines to illustrate the difference between actual and predicted values.

```
import matplotlib.pyplot as plt
numcourses = [13,4,12,3,14,13,12,9,11,7,13,11,9,2,5,7,10,0,9,7]
happiness  = [70,25,54,21,80,68,84,62,57,40,60,64,45,38,51,52,58,21,75,70]
plt.figure(figsize=(6,6))
plt.plot(numcourses,happiness,'ks',markersize=15)
plt.xlabel('Number of courses taken')
plt.ylabel('General life happiness')
plt.xlim([-1,15])
plt.ylim([0,100])
plt.grid()
plt.xticks(range(0,15,2))
plt.savefig('Figure_11_03.png',dpi=300)
plt.show()
# Build a statistical model
# design matrix as a column vector
X = np.array(numcourses,ndmin=2).T
print(X.shape)
# fit the model using the left-inverse
X_leftinv = np.linalg.inv(X.T@X) @ X.T
# solve for the coefficients
beta = X_leftinv @ happiness
beta
# predicted data
pred_happiness = X@beta
plt.figure(figsize=(6,6))
# plot the data and predicted values
plt.plot(numcourses,happiness,'ks',markersize=15)
plt.plot(numcourses,pred_happiness,'o',color=[.6,.6,.6],linewidth=3,markersize=8)
# plot the residuals (errors)
for n,y,yHat in zip(numcourses,happiness,pred_happiness):
  plt.plot([n,n],[y,yHat],'--',color=[.8,.8,.8],zorder=-10)
plt.xlabel('Number of courses taken')
plt.ylabel('General life happiness')
plt.xlim([-1,15])
plt.ylim([0,100])
plt.xticks(range(0,15,2))
plt.legend(['Real data','Predicted data','Residual'])
```

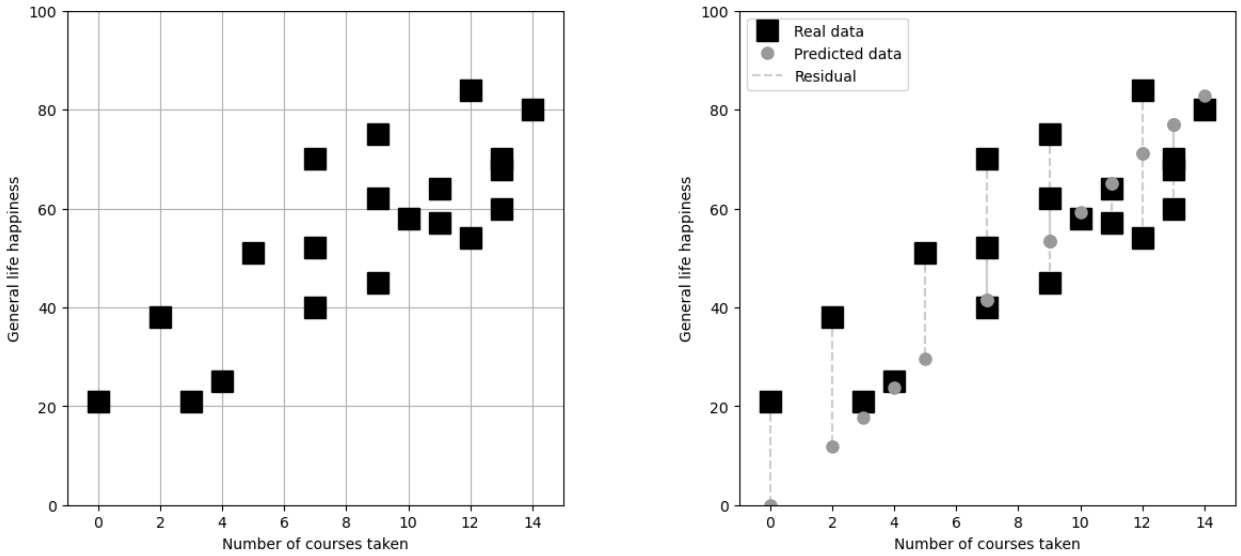


Figure 2-Outputs

Linear Regression Using Left-Inverse and QR Decomposition

This script calculates the regression coefficients (betas) for the relationship between the number of courses taken and general life happiness using two methods: the left-inverse and QR decomposition. The design matrix X includes a column of ones for the intercept, while the response vector y contains the happiness values. The left-inverse method is used to compute the coefficients by solving $\beta = (X^T \cdot X)^{-1} X^T y$ and the QR decomposition method is applied to solve for the coefficients using $R^{-1} Q^T y$. The betas from both methods are printed and compared for verification.

```
import numpy as np
import matplotlib.pyplot as plt
# null space
from scipy.linalg import null_space
import sympy as sym
numcourses = [13,4,12,3,14,13,12,9,11,7,13,11,9,2,5,7,10,0,9,7]
happiness  = [70,25,54,21,80,68,84,62,57,40,60,64,45,38,51,52,58,21,75,70]
# recreate the design matrix and solution via left-inverse
X = np.hstack((np.ones((20,1)),np.array(numcourses, ndmin=2).T))
y = np.array(happiness, ndmin=2).T
# Left-inverse method
beta1 = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
# QR decomposition
Q, R = np.linalg.qr(X)
beta2 = np.dot(np.linalg.inv(R), np.dot(Q.T, y))
print('Betas from left-inverse: ')
print(np.round(beta1, 3))
```

```
print(' ')
print('Betas from QR with inv(R): ')
print(np.round(beta2, 3))
print(' ')
```