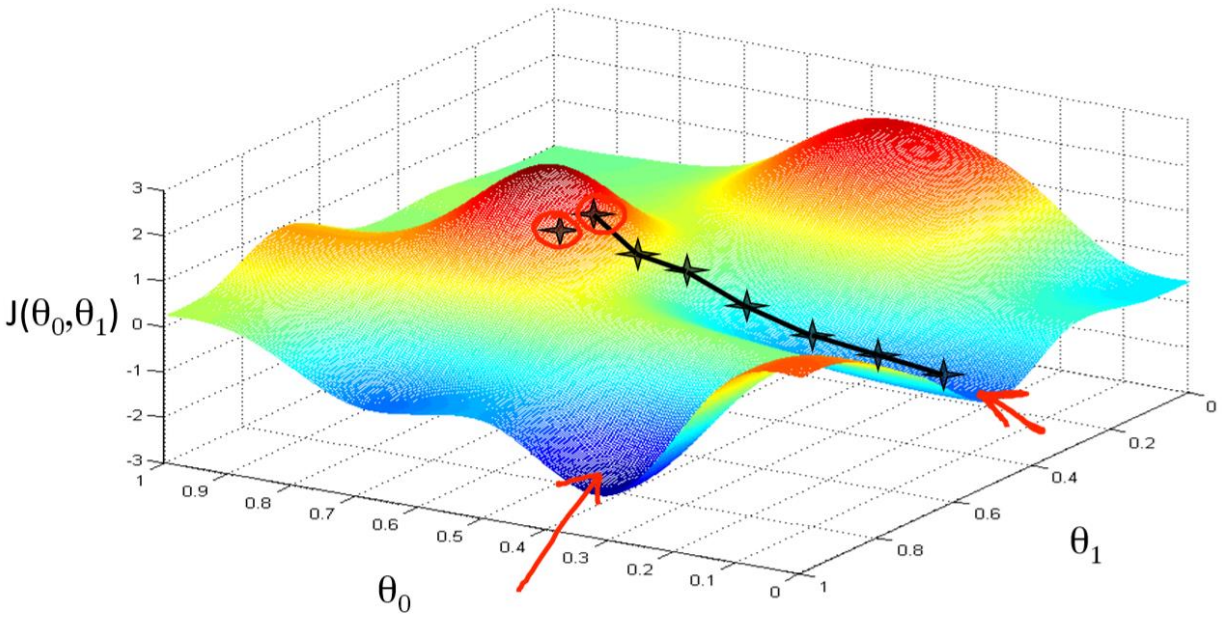
Ben-Gurion University of the Negev

Mathematical Methods for Data Science

August 2020

Lecturer: Prof. Jonathan D. Rosenblatt

MSc Student: Tom Landman



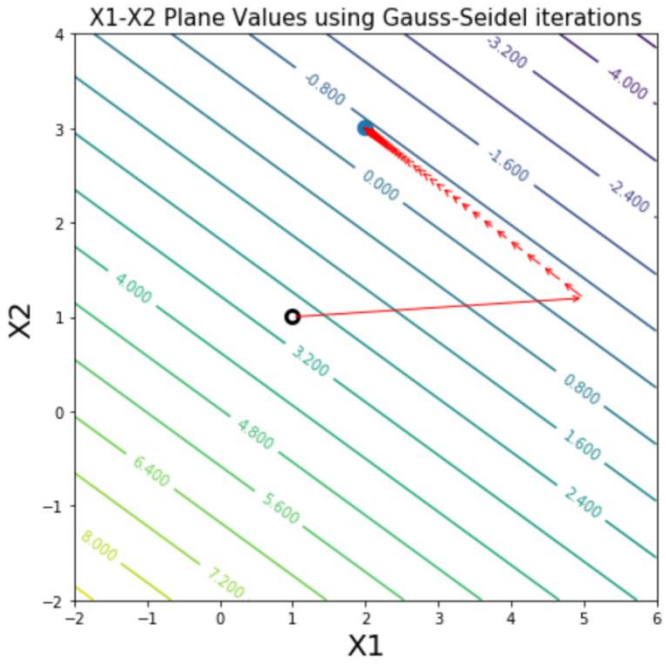
Answers

1. In the following, supply the code, and a plot, of the iterations in the x_1, x_2 plane, starting at $x = (1, 1)$. Define $Ax = b$ where

$$A := \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix};\ b := (13, 21)$$

1.1. Find x using Gauss-Seidel iterations.

Based on Golub, Gene H., et al. book [1], in Gauss-Seidel method for solving $Ax = b$, the matrix A is split into 2 parts: U, an upper triangular matrix similar to A upper triangulae elements; L, a lower triangular matrix which is calculated using the following formula: $L = A - U$. This can be converted into an iterative method by the following recurrence: $x_{(i+1)} = L^{-1}(b - Ux_i)$. Further, the function *GS(...)* which described below was developed using the abovementioned equations, and the function *plotChart()* was developed in order to plot the iterations in the x_1, x_2 plane for all of the below mentioned iterative methods. Further, by plugging in the given arguments of A, x_0, b, the most accurate x values, which are [2 3] achieved after 199 iterations using $\varepsilon = 10E - 11$, and resulted in the following x_1, x_2 plane plot which described below.



1.2. Find x using Jacobi iterations.

Based on Shewchuk, J.R., paper [2] the Jacobi Method for solving $Ax = b$, the matrix A is splitted into 2 parts: D, whose diagonal elements are identical to the elements of A, and the rest are zero; E, whose off diagonal elements are equal to the elements of A, while the diagonal elements are zero. Therfore,

$$A = D + E$$

The Jacobi method is derived as follows:

$$Ax = b$$

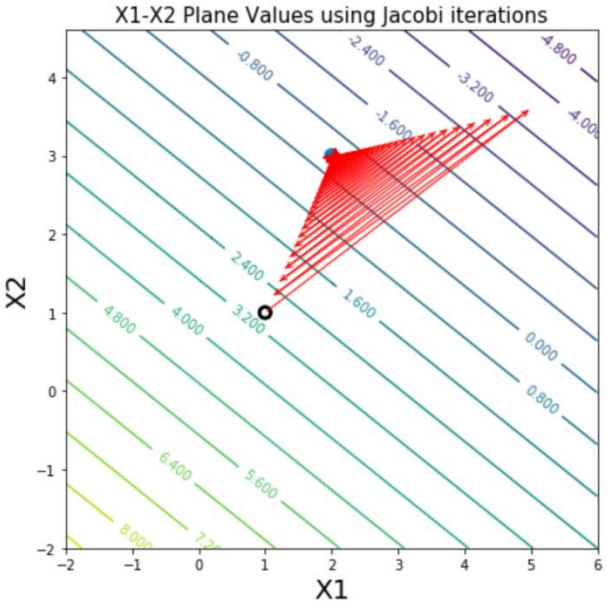
$$Dx == -Ex + b$$

$$x = -D^{-1}Ex + D^{-1}b$$

$$x = Bx + z, \qquad \text{where } B = -D^{-1}E, \qquad z = D^{-1}b$$

This can be converted into an iterative method by the following recurrence: $x_{(i+1)} = Bx_i + z$.

Further, the function *jacobi(…)* which described below was developed using the abovementioned equations. Further, by plugging in the given arguments of A, x_0, b, the most accurate x values, which are [2 3], achieved after 387 iterations using $\varepsilon = 10E - 9$, and resulted in the following $x_1, x_2\ plane$ plot which described below.



1.3. Find x using Steepest Descent (a.k.a. Gradient Decent), with exact line-searches.

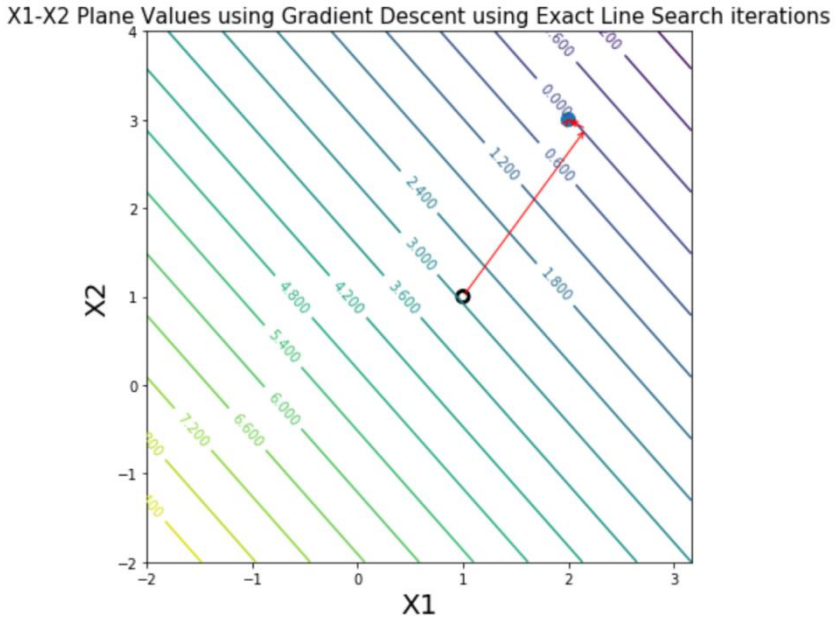
Based on Shewchuk, J.R., paper [2], the required equations for solving $Ax = b$ system using Gradient Descent with exact line-searches are as follows

$$r_i = b - Ax_i$$

$$\alpha_i = \frac{r_i^T r_i}{r_i^T A r_i}$$

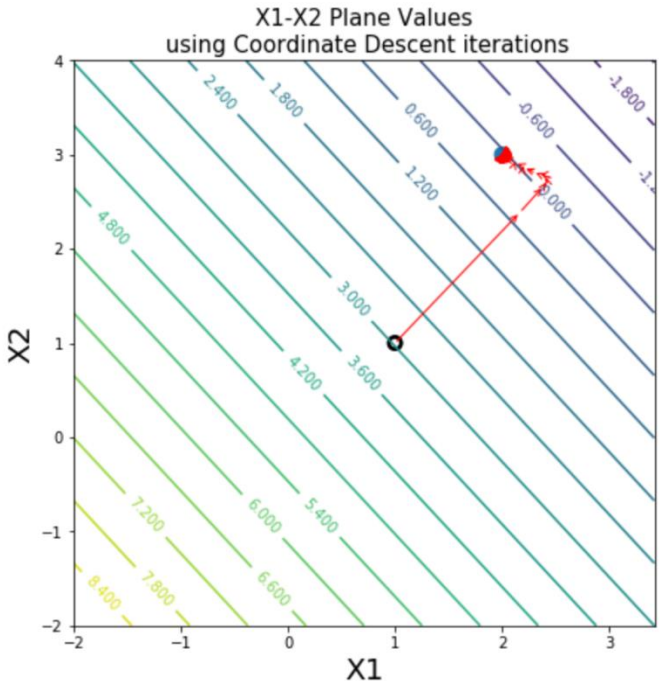
$$x_{(i+1)} = r_i + \alpha_i r_i$$

Further, the function *GDELS(…)* which described below was developed using the abovementioned equations. Further, by plugging in the given arguments of A, x_0, b, the most accurate x values, which are [2 3], achieved after 5 iterations using $\varepsilon = 10E - 6$, and resulted in the following x_1, x_2 *plane* plot which described below. Furthermore, this method outperformed the rest of the optimization methods.



1.4. Find x using coordinate-decent, i.e., optimizing a single coordinate per iteration. You are free to choose the type of descent you perform at each coordinate.

Unlike Gradient Descent, Based on Shewchuk, J.R., paper [2], Coordinate Descent is a method for optimizing each coordinate separately. Moreover, the function *CD(…)* which described below was developed similar to the Gradient Descent algorithm that was presented above, however, I added an internal loop in order to optimize each coordinate separately. Further, by plugging in the given arguments of A, x_0, b, the most accurate x values, which are [2 3], achieved after 124 iterations using $\varepsilon = 10E - 11$, and resulted in the following x_1, x_2 *plane* plot which described below.



2. Generate 1000 samples from $\boldsymbol{Binom(10, 0.5)}$. You are allowed to use your software's *Unif*[0, 1] generator. I want the code and a histogram of 1,000 samples.

2.1. Use the inverse probability transform ($\boldsymbol{F^{-1}(t)}$). You are allowed to use your software's quantile functions.

First, denote the binomial probability function as $X \sim Binomial(n,p) \rightarrow P_k = P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$. Second, based on P. Robert et al. book [3] which describes the Inverse Transform method, the binomial inverse probability transform ($F^{-1}(t)$) needed to be found , by calculating the increment between two consecutive probabilities where

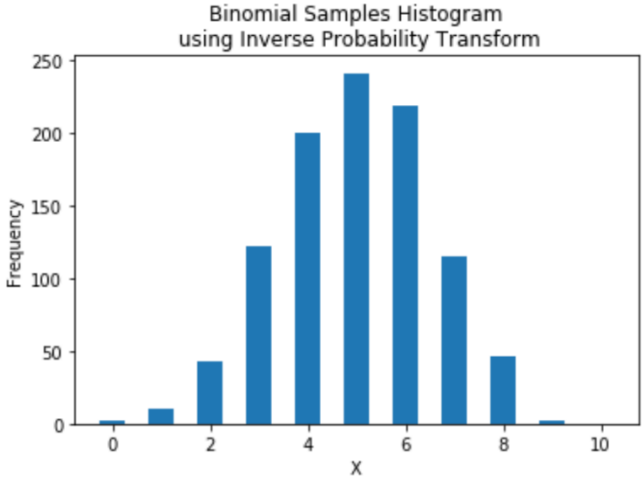
$$\frac{P_{k+1}}{P_k} = \frac{\binom{n}{k+1} p^{k+1}(1-p)^{n-k-1}}{\binom{n}{k} p^k (1-p)^{n-k}} = \frac{\binom{n}{k+1}}{\binom{n}{k}} \cdot \frac{p}{(1-p)}$$

$$\frac{\binom{n}{k+1}}{\binom{n}{k}} = \frac{n!}{(n-k-1)!\,(k+1)!} \frac{(n-k)!\,k!}{n!} = \frac{n!}{(n-k-1)!\,(k+1)k!} \frac{(n-k)(n-k-1)!\,x!}{n!} = \frac{n-k}{k+1}$$

By combining the abovementioned equations it can be shown that

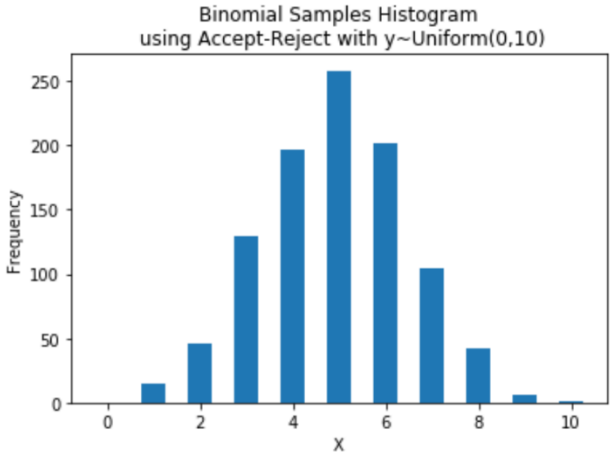
$$\frac{P_{k+1}}{P_k} = \frac{\binom{n}{k+1}}{\binom{n}{k}} \cdot \frac{p}{(1-p)} = \frac{n-k}{k+1} \cdot \frac{p}{(1-p)} \Rightarrow P_{k+1} = \frac{n-k}{k+1} \cdot \frac{p}{(1-p)} P_k$$

Finally, in order to generate a binomial random variable X, the inverse probability method needed to be used with the following parameters: k, which represents the possible value of X, P which is the probability that $X = k$, and F is the probability that $X \leq k$ where $F = \sum_{j=0}^{k} P_j$. Further, both samples generation and below histogram of 1000 binomial samples were calculated using the code below which describes the implementation of this method.



2.2. Use accept reject (a.k.a. rejection sampling), with proposals from $\boldsymbol{Unif}\{\mathbf{0}, \mathbf{10}\}$. Use only your software's $\boldsymbol{Unif}[\mathbf{0}, \mathbf{1}]$ generator.

Note that since the accept reject method requires a normalization factor denoted as $\boldsymbol{M}$ which is relevant to the acceptance condition $U \leq \frac{f(y)}{m \cdot g(y)}$, an appropriate value that fulfills the following condition: $f(x) \leq M \cdot g(x)$ needed to be found. Therefore, by simulating the binomial probability space where $n = 10, p = 0.5$ which resulted in finding the maximal value of $P_{max} = P(x = 5 | n = 10, p = 0.5) = 0.24609375$. Then, $\boldsymbol{M}$ was found by multiplying P_{max} with $g(x) = \frac{1}{10}$, since $g(x) \sim Uniform(0,10)$ which resulted in the following: $M = P_{max} \cdot g(x) = 0.24609375 \cdot 0.1 \cong 2.4609375$. Further, this method was implemented using the *binomial_AR()* function which described below and based on P. Robert et al. book [3]. Moreover, since $\boldsymbol{M}$ depends on the given probability and the number of trials it was designed to address the general case where $\boldsymbol{n}$ and $\boldsymbol{p}$ are unknown. Furthermore, the histogram below was generated using the *binomial_AR()* function.



2.3. Use Metropolis-Hastings with a proposal distribution of your choice.

Based on P. Robert et al. book [3], the Metropolis-Hastings Algorithm steps are as follows:

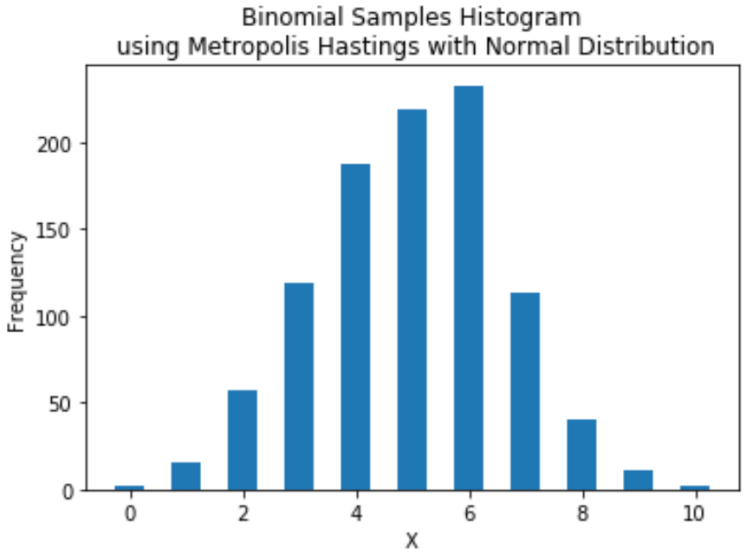
Given x_t,

1. *Generate* $Y_t \sim q(y/x_t)$
2. *Take* $X_{t+1} = \begin{cases} Y_t \ \ with\ probability \ \ \rho(x_t, Y_t), \\ x_t \ \ with\ probability \ \ 1-\rho(x_t, Y_t), \end{cases}$

Where

$$\rho(x,y) \ = \ min\left\{\frac{f(y)}{f(x)}\frac{q(x|y)}{q(y|x)}, 1\right\}$$

Note that in the case of choosing a symmetric proposal density, the proposal density ratio $\frac{q(x|y)}{q(y|x)}$ which goes in two directions is equal to 1. Therefore, I chose to use the normal distribution as the proposed density, which is symmetric and moreover, in order to generate values between [0,10] I used the following parameters $Y_t \sim N(\mu = 5, \ \sigma = \mu/3)$. Further, note that since only a $uniform(0,1)$ random variable generation function is allowed to be used, first, I used the Box-Muller transform to generate $z \sim N(\mu = 0, \sigma = 1)$. Secondly, I shifted and scaled z to the appropriate values using $Y_t \ = \ z \cdot \sigma \ + \mu, where \ \ \mu = 5, \ \sigma = \mu/3$. Finally, as described below, I have implemented this method using the *normal_BM()* function to generate a normal distributed random variable and the *binomial_MH()* function which contain the Metropolis-Hastings algorithm implementation and a histogram generation.



3. What are the first 10 numbers in the sequence of a linear congruent generator with:

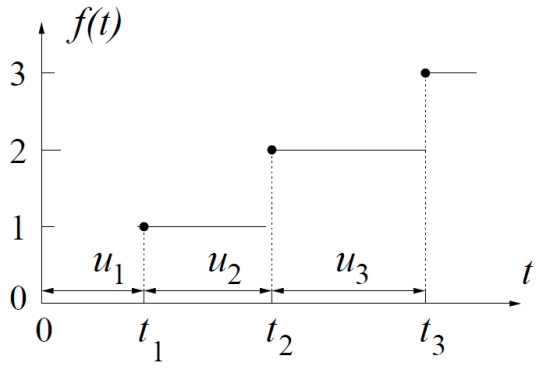
a = 1664525, c = 0, m = 232 , X_0 = 3? Provide sequence and code.

The requested generated sequence was generated using the code below that was written based on Tezuka S. book, Uniform Random Numbers Theory and Practice [4].

$$x_1 = 4993575,\ x_2 = 1168709115, x_3 = 232464319,\ x_4 = 476952243,\ x_5 = 4292385047,$$
$$x_6 = 1044246571,\ x_7 = 1258902575,\ x_8 = 3214606435,\ x_9 = 2259777287,\ x_{10} = 740218203$$

4. Let N_t be a simple birth process, i.e., a Poisson point process with rate $\lambda_i = \lambda i$. Let X_i be the times between event, so that $N_t = max\{n\ s.t. \sum_{i=o}^{n} X_i \leq t\}$. Write the likelihood of λ, given $X_1, \ldots, X_N$. Is it convex in λ?

Based on Bruce Hajek notes for ECE 534 [5], a function $f \in \mathbb{R}_+$ is called a *counting function* based on the following conditions: $f(0) = 0, f\ is\ nondecreasing, f\ is\ right\ continous\ and\ integer\ valued$. The number of counts during an interval of $(o, t]$ denoted as $f(t)$. In addition, an increment defined as the number of counts in the interval $(a, b]$ is denoted as $f(b) - f(a)$. The time for the ith count for $i \geq 1$ denoted as *count time* t_i, which results in describing f as a sequence of t_i. Further, f can be also described as a sequence of *intercount times* u_i, if $u_1 = t_1$ and $u_i = t_i - t_{i-1}\ for\ i \geq 2$. A graphical demonstration of count times and intercount times sequences is described below.



Further, by definition, a *Poisson Process* with rate of $\lambda > 0$ is a random process $N = (N_t : t \geq 0)$ such that $N\ is$ a counting process with independent increments. In addition, $N(t) - N(s)$ has the $Pois(\lambda(t-s))$ distribution for $t \geq s$. As a result, $u_1, u_2, \ldots, u_n$ are mutually independent $Exp(\lambda)$ random variables. Recall that the exponential distribution density function is

$$f(x|\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Based on [6] In order to calculate λ likelihood function given $X_1, X_2, \ldots, X_n$ I'll use the fact that $X_1, X_2, \ldots, X_n$ are distributed exponentially with rate of $\lambda_i = \lambda i$, the likelihood function will be written as follows:

$$L(\lambda|X_1, X_2, \ldots, X_n) = \prod_{i=1}^{\boldsymbol{N_t = max\{n\ s.t. \sum_{i=o}^{n} X_i \leq t\}}} \lambda_i e^{-\lambda_i x_i} = \prod_{i=1}^{\boldsymbol{N_t = max\{n\ s.t. \sum_{i=o}^{n} X_i \leq t\}}} \lambda i e^{-\lambda i x_i} = \lambda^n n!\, e^{-\lambda \sum_{i=1}^{n} i x_i}$$

The log-likelihood function will be written as follows:

$$l(\lambda|X_1, X_2, \ldots, X_n) = ln(\lambda^n n!\, e^{-\lambda \sum_{i=1}^{n} i x_i}) = nln(\lambda) + ln(n!) - \lambda \sum_{i=1}^{n} i x_i$$

Finally, in order to find the maximum likelihood estimator for λ I'll take the derivative of the log-likelihood function to be equal to zero as follows:

$$\frac{dl(\lambda|X_1, X_2, \ldots, X_n)}{d\lambda} = \frac{dl(nln(\lambda) + ln(n!) - \lambda \sum_{i=1}^{n} i x_i)}{d\lambda} = \frac{n}{\lambda} - \sum_{i=1}^{n} i x_i = 0$$

$$\frac{n}{\lambda} = \sum_{i=1}^{n} i x_i \xrightarrow{yields} \hat{\lambda} = \frac{n}{\sum_{i=1}^{n} i x_i}$$

In order to check If $\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} i x_i}$ is convex I'll use the definition of convex function based on Boyd, S., et al. book, Convex Optimization [7]. A function $f: \mathbb{R}^n \to \mathbb{R}$ is *convex* if $\boldsymbol{dom}\ f$ is a convex set and if for all x, $y \in \boldsymbol{dom}\ f$, and $\theta\ with\ 0 \leq \theta \leq 1$, $f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$

Further, geometrically speaking, it means that the line segment between $(x, f(x))\ and\ (y, f(y))$, denoted as the *chord* between $x\ and\ y$ lies above the graph of f. In the following case, we have $\hat{\lambda} = f(x) = \frac{n}{\sum_{i=1}^{n} i x_i}$, by plugging $f(x)$ in the above inequality we get the following:

$$f(\theta x + (1-\theta)y) = \frac{n}{\sum_{i=1}^{n} i(\theta x_i + (1-\theta) y_i)} = \frac{n}{\sum_{i=1}^{n} i(\theta x_i + y_i - \theta y_i)}$$

$$\theta f(x) + (1-\theta) f(y) = \theta \frac{n}{\sum_{i=1}^{n} i x_i} + (1-\theta) \frac{n}{\sum_{i=1}^{n} i y_i}$$

Since $x_i, y_i > 0$, for this proof I'll assume that both $\forall\ i\ x_i = y_i = 1$ which results in the following:
First, $f(x) = \frac{n}{\sum_{i=1}^{n} i x_i} = \frac{n}{\sum_{i=1}^{n} i} = f(y)$. Second, since the accumulation at the denominator is arithmetic series it can be calculated using this formula $S_n = \frac{n(a_1 + a_n)}{2}$. Further, under the assumptions, the value of $f(x)$ is equal to $f(x) = \frac{n}{\sum_{i=1}^{n} i} = \frac{n}{\frac{n(n+1)}{2}} = \frac{2}{n+1}$. Finally, by plugging this into the inequality it can be shown that

$$(\theta x + (1-\theta)y) = \frac{n}{\sum_{i=1}^{n} i(\theta x_i + (1-\theta)y_i)} = \frac{n}{\sum_{i=1}^{n} i(\theta x_i + y_i - \theta y_i)} = \frac{n}{\sum_{i=1}^{n} i(\theta + 1 - \theta)} = \frac{2}{n+1}$$

$$\theta f(x) + (1-\theta)f(y) = \theta \frac{n}{\sum_{i=1}^{n} i x_i} + (1-\theta)\frac{n}{\sum_{i=1}^{n} i y_i} = \theta \frac{n}{\sum_{i=1}^{n} i} + (1-\theta)\frac{n}{\sum_{i=1}^{n} i}$$

$$= \theta \frac{2}{n+1} + (1-\theta)\frac{2}{n+1} = \frac{2}{n+1}$$

Therefore, since the inequality is now confirmed $f(x)\ is\ convex$ since,

$$f(\theta x + (1-\theta)y) = \frac{2}{n+1} \leq \frac{2}{n+1} = \theta f(x) + (1-\theta)f(y)$$

However, in the general case where $x_i\ and\ y_i$ are unknown, by plugging in the extreme values of θ where $\theta = 0\ or\ \theta = 1$ we get equality for both sides, which means that $f(x)$ is not *strictly convex.*

5. Prove that the leading eigenvalue of Markov Chain's transition matrix is 1.

Based on the book of Guttorp, P. and N. Minin, V. [8] Let A_{nxn} be a Markov Chain's transition matrix, which is a left stochastic matrix such that each entry p_{ij} is non-negative and moreover, the sum of each column entries is 1. In order to simplify this proof, I'll use the fact that for a squared matrix, its determinant is equal to its transposed matrix determinant:
$det(A) = det(A^T)$. Therefore, after transposing A, it can be treated as a right stochastic matrix where $a_{ij} \geq 0$ and $a_{i1} + a_{i2} + \ldots + a_{in} = 1$. In addition, by def., λ is an eigenvalue of A' if and only if $A'v = \lambda v$ for some **nonzero vector v**. By letting v be a vector of ones the following equation can be derived:

$$eq.1.\ A'v = \begin{pmatrix} a_{i1} & \cdots & a_{in} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{n} a_{ij} \\ \vdots \\ \sum_{j=1}^{n} a_{nj} \end{pmatrix} \overset{\text{right stochastic def.}}{=} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix};\quad eq.2.\ \lambda v = \lambda \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

By the eigenvalue def. and based on the above equations it can be derived that the eigenvalue is 1:

$$eq.3.\ A'v = \lambda v \ \rightarrow \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 1 \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \rightarrow \lambda = 1$$

Now to show that the absolute value of any eigenvalue of the stochastic matrix A is less than or equal to 1. First, since λ is an eigenvalue of A if and only if $Av = \lambda v$ for some **nonzero vector v**, by comparing the $i - th$ row of both sides, for each $1 \leq i \leq$ n the following equation can be derived

$$eq.4.\ a_{i1}v_1 + a_{i2}v_2 + \cdots + a_{in}v_n = \lambda v_i$$

Second, let v_k be the entry of v that has the maximal absolute value;

$$eq.5.\quad |v_k| = max|v_1|, |v_2|, \ldots, |v_n|$$

Note that $|v_k| > 0$, otherwise $v = 0$ which contradicts that an eigenvector is a nonzero vector. As a result, using eq.4. it can be shown that with $i = k$,

$$
\begin{aligned}
eq.6.\ \ |\lambda||v_k| &= |a_{k1}v_1 + a_{k2}v_2 + \cdots + a_{kn}v_n| \\
&\le a_{k1}|v_1| + a_{k2}|v_2| + \ldots + a_{kn}|v_n| \quad (\textit{by the triangle inequality and } a_{ij} \ge 0) \\
&\le a_{k1}|v_k| + a_{k2}|v_k| + \ldots + a_{kn}|v_k| \quad (\textit{since } |v_k| \textit{ is the maximal}) \\
&= (a_{k1} + a_{k2} + \ldots + a_{kn})|v_k| = |v_k|
\end{aligned}
$$

Finally, since $|v_k| > 0, it\ follows\ that\ \lambda \le 1$.

6. Show that the regression's "Hat Matrix" ($\boldsymbol{H = (X'X)^{-1}X'}$ is the Moore-Penrose Pseudo-Inverse of the matrix X.

The true regression model is defined as $\boldsymbol{Y = X\beta + \varepsilon}$ while $\boldsymbol{X}$ defined as the matrix of independent variables observations, $\boldsymbol{Y}$ as the vector of dependent variable observations, $\boldsymbol{\beta}$ the true coefficient vector, and a noise term denoted as $\boldsymbol{\varepsilon}$. The predicted model defined as $\boldsymbol{\widehat{Y} = X\widehat{\beta}}$ while $\boldsymbol{\widehat{\beta}}$ denoted as the estimated coefficient vector. Since the sum of squared errors defined as $\boldsymbol{SSE = (Y - X\beta)'(Y - X\beta)}$, it can be minimized by calculating the partial derivatives for each $\boldsymbol{\beta_i}$ and moreover, set it to be equal to zero, the Normal equations are derived, which result in the following equation:

$$
\boldsymbol{(X'X)\beta = (X'Y) \longrightarrow (X'X)^{-1}(X'X)\beta = (X'X)^{-1}(X'Y) \longrightarrow I\beta = (X'X)^{-1}(X'Y)}
$$

$$
\boldsymbol{\longrightarrow \widehat{\beta} = (X'X)^{-1}X'Y \longrightarrow \widehat{\beta} = HY}\ (H\ defined\ as\ the\ regression\ Hat\ Matrix)
$$

In addition, based on Golub, Gene H., and Charles F. Van Loan book [1]note that the pseudo inverse (or Moore-Penrose inverse) of a matrix $\boldsymbol{A}$ is the matrix $\boldsymbol{A^+}$ that fulfills the following conditions:

I. $\boldsymbol{AA^+A = A}$
II. $\boldsymbol{A^+AA^+ = A^+}$
III. $\boldsymbol{AA^+}$ $\boldsymbol{symmetric}$ ($\boldsymbol{A}$ symmetric if $\boldsymbol{A}$=$\boldsymbol{A'}$)
IV. $\boldsymbol{A^+A}$ $\boldsymbol{symmetric}$

In addition, in case when $\boldsymbol{A}$ has linearly independent columns (and thus the matrix $\boldsymbol{A'A}$ is invertible), $\boldsymbol{A^+}$ can be computed as $\boldsymbol{(A'A)^{-1}A'}$.

In order to prove that $\boldsymbol{H}$ is the Moore-Penrose Pseudo-Inverse of the matrix $\boldsymbol{X}$, the abovementioned conditions needed to be proofed:

I. $\boldsymbol{XHX = X((X'X)^{-1}X')X = X(X'X)^{-1}X'X = XI = X = XX^+X}$

II. $\boldsymbol{HXH = ((X'X)^{-1}X')X((X'X)^{-1}X') = (X'X)^{-1}X'X(X'X)^{-1}X' = I(X'X)^{-1}X' = X^+ = X^+XX^+}$

III. $\boldsymbol{(XX^+)' = (XH)' = H'X' = ((X'X)^{-1}X')'X' = X(X'X)^{-1}X' = XH = XX^+ \longrightarrow XX^+}$ $\boldsymbol{is\ symmetric}$

IV.$(\boldsymbol{X^{+}X})' = (\boldsymbol{HX})' = \boldsymbol{X'H'} = \boldsymbol{X'}((\boldsymbol{X'X})^{-1}\boldsymbol{X'})' = \boldsymbol{X'}(\boldsymbol{X}(\boldsymbol{X'X})^{-1}) = \boldsymbol{X'X}(\boldsymbol{X'X})^{-1} = \boldsymbol{I}\ (\boldsymbol{identity\ is\ symmetric}) \longrightarrow \boldsymbol{X^{+}X\ is\ symmetric}$

As a result, by approving all of the conditions above, it can be inferred that $\boldsymbol{X^{+}} = (\boldsymbol{X'X})^{-1}\boldsymbol{X'} = \boldsymbol{H}$ meaning that $\boldsymbol{H}$ the regression's "Hat Matrix" is the Moore-Penrose Pseudo-Inverse of the matrix $\boldsymbol{X}$.

7. Prove that the QR decomposition of a matrix may be found with a series of Householder Transformations. How many floating-point operations (FLOPS) are required (explain)?

Based on Golub, Gene H., and Charles F. Van Loan book, Matrix Computations [1], QR decomposition defined as a rectangular matrix $\boldsymbol{A} \in \mathbb{R}^{mxn}$ which can be decomposed into a product of an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{mxm}$ $and\ an\ upper\ triangular\ \boldsymbol{R} \in \mathbb{R}^{mxn}$ so that $\boldsymbol{A} = \boldsymbol{QR}$. **I**n general, there are several ways to perform QR decomposition, and in particular, the Householder QR method will be described here. First, a brief explanation regarding Householder reflections will be presented. Second, I'll describe the QR algorithm using Householder transformations and last, calculations regarding how many floating-point operations (*FLOPS)* are required during this technique will be presented.

First, since orthogonal matrices serve an important role when dealing with eigenvalues, least squares and other calculations, recall that $\boldsymbol{Q} \in \mathbb{R}^{mxm}$ defined as an orthogonal matrix if $\boldsymbol{Q^{T}Q} = \boldsymbol{QQ^{T}} = \boldsymbol{I_{m}}$. Second, in order to explain what Householder Reflections are and to understand rotations and reflections associated geometry, a demonstration in m=2 level taken from [1] will be given. Note that orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{2x2}$ is defined as *rotation* if it has the form:

$$Q = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}$$

In the case of $y = Q^{T}x$, x is rotated counterclockwise through angle θ which results in *y*. Further, Note that orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{2x2}$ is defined as a *reflection* if it has the form:

$$Q = \begin{bmatrix} cos(\theta) & sin(\theta) \\ sin(\theta) & -cos(\theta) \end{bmatrix}$$

In the case of $y = Q^{T}x = Qx$, x is reflected across the line is defined as $S = span\left\{\begin{bmatrix} cos(\theta/2) \\ sin(\theta/2) \end{bmatrix}\right\}$ which results in *y*. Moreover, since both rotations and reflections can be constructed easily and can contain zeros within vectors by choosing the proper angle or the reflection plane, they are computationally attractive.

A *Householder reflection* defined as follows: Let $v \in \mathbb{R}^{m}$ be a nonzero vector denoted as the Householder vector, which is usually normalized i.e. $||v|| = 1$, in order to simplify calculation ($v^{T}v = 1$) and to permit the storage of $v(2{:}m)$ where the zeros are introduced in x meaning $only\ with\ rank - 1$. $P \in \mathbb{R}^{mxm}$ is defined as Householder reflection if it has the form $P = I_{m} - \beta vv^{T}$ where $\beta = \frac{2}{v^{T}v}$. In case vector x is multiplied by the householder matrix P, it is reflected in the hyperplane $span\{v\}^{\perp}$ as follows:

$$Px = (I_m - \beta vv^T)x = x - \beta vv^T x$$

In addition, note that P is symmetric and orthogonal, which explained below:

$$P^T = (I_m - \beta vv^T)^T = I_m - \beta vv^T = P \rightarrow \ P\ is\ symmetric$$

$$P^T P = PP = (I_m - \beta vv^T)(I_m - \beta vv^T) = I_m - \beta vv^T - \beta vv^T + \beta\beta vv^T vv^T = I_m - 2\beta vv^T + \beta\beta vv^T vv^T =$$
$$I_m - 2\frac{2}{v^T v} vv^T + \frac{2}{v^T v}\frac{2}{v^T v} vv^T vv^T \ = I_m - 4vv^T + 4vIv^T \ P = \ I_m \ (since\ ||v|| = 1) = PP^{-1}$$
$$\rightarrow \ P\ is\ \text{orthogonal}$$

Given $A \in \mathbb{R}^{mxn}$ with $m \geq n$, the Housholder QR algorithm finds Householder matrices $H_1, H_2, \ldots, H_n$ such that in the case of $Q = H_1 \cdot H_2 \cdot \ldots \cdot H_n$, then $Q^T A = R$ is upper triangular. In addition, note that the upper triangular part of A is overwritten by the upper triangular part of R and components $j + 1{:}\, m$ of the jth Householder vector are stored in $A(j + 1{:}\, m, j), j < m$.

$For\ j = 1{:}\, n$
  $[v, \beta] \ = \ house(A(j{:}\, m, j))$
  $A(j{:}\, m, j{:}\, n) \ = \ (I - \beta vv^T)A(j{:}\, m, j{:}\, n)$
  if $j < m$
    $A(j + 1{:}\, m, j) \ = \ v(2{:}\, m - j + 1)$
  end
end

Further, as described above, after n steps, an upper triangular R is constructed such that $R = H_n H_{n-1} \cdots H_1 A$ and by setting $Q = H_1 \cdot H_2 \cdots H_n$, $A = QR$ is obtained. Furthermore, the Housholder QR algorithm requires $2n^2(m - n/3)$ flops, and since the matrix $Q = H_1 \cdot H_2 \cdots H_n$ also required, it can be accumulated using $4(m^2 n \ - \ mn^2 + n^3/3)$ flops based on the following accumulation algorithm:

$Q = I_m(:, 1{:}\, k)\ where\ 1 \leq k \leq m$
$For\ j = n{:}\, -1{:}\, 1$
  $v(j{:}\, m) \ = \ \begin{bmatrix} 1 \\ A(j + 1{:}\, m, j) \end{bmatrix}$
  $\beta_j \ = \ 2/(1/+\|A(j + 1{:}\, m, j)\|_2^2$
  $Q(j{:}\, m, j{:}\, k) \ = \ Q(j{:}\, m, j{:}\, k) \ - (\beta_j v(j{:}\, m)(v(j{:}\, m)^T Q(j{:}\, m, j{:}\, k))$
end

As a result, in order to calculate both $Q\ and\ R$ matrices, we need approximately $\frac{4n^3}{3}$ flops.

Reference

[1] Golub, Gene H. and C. F. Van Loan., *Matrix computations*, Vol. 4. 2013.

[2] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," 1994.

[3] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, vol. 102. 2006.

[4] S. Tezuka and S. Tezuka, "Linear Congruential Generators," in *Uniform Random Numbers*, Springer US, 1995, pp. 57–82.

[5] B. Hajek, "An exploration of random processes for engineers," *Cl. notes ECE*, vol. 534, pp. 1–405, 2011.

[6] D. R. Cox and D. V. Hinkley, *Theoretical statistics*. Chapman and Hall/CRC, 1974.

[7] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. United States of America by Cambridge University Press, New York, 2004.

[8] P. Guttorp and V. N. Minin, *Stochastic Modeling of Scientific Data*. Chapman and Hall/CRC, 2018.