1 (a) We know that at time $k=4$, $X[4]=-1$, $Z[4]=1$

So $\Pi_X[4]=\begin{bmatrix}0\\1\end{bmatrix}$ $\Pi_Z[4]=\begin{bmatrix}1\\0\end{bmatrix}$, where $\Pi_{X_1}(k)$, $\Pi_{Z_1}(k)$ means the probability of $X, Z=1$ at time k, $\Pi_{X_2}(k)$, $\Pi_{Z_2}(k)$ is the probability of $X, Z=-1$ at time k.

According to chapman-kolmogorov Equ,

$$\begin{cases}\Pi_X[k+1]=A_1'\,\Pi_X[k]\\ \Pi_Z[k+1]=A_2'\,\Pi_Z[k]\end{cases}$$

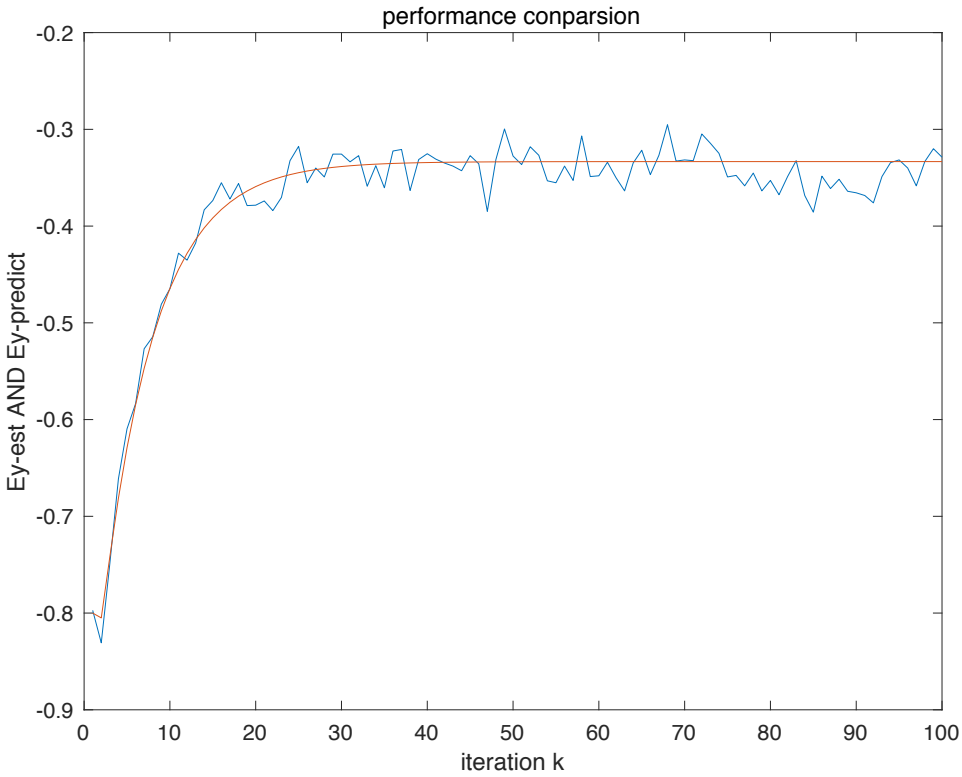
$$\Rightarrow\begin{cases}\Pi_X[k]=A_1'^{k-4}\,\Pi_X[4]\\ \Pi_Z[k]=A_2'^{k-4}\,\Pi_Z[4]\end{cases}$$

So, $E\{Y[k]=X[k]+Z[k]\mid X[4]=-1, Z[4]=1\}$

$=(\Pi_{X_1}(k)+\Pi_{Z_1}(k))\times 1+(\Pi_{X_2}(k)+\Pi_{Z_2}(k))\times(-1)$

Where, $\Pi_X[k]=\begin{bmatrix}\Pi_{X_1}(k)\\ \Pi_{X_2}(k)\end{bmatrix}$ $\Pi_Z[k]=\begin{bmatrix}\Pi_{Z_1}(k)\\ \Pi_{Z_2}(k)\end{bmatrix}$

(b) We draw the estimation of Markov chain's real value of $E\{Y\}$ and the estimate value of our function. We can see that after a few iterations, the true value will jump up and down our function value.


performance conparsion
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9
Ey-est AND Ey-predict
0
10
20
30
40
50
60
70
80
90
100
iteration k

2.

2<1> We have P_0 and $\hat{P}_0$ which are two covariance matrix estimates of x_0.

And P_0 and $\hat{P}_0$ are positive-definite matrix

Our process $x[k]$ defined as $x[k+1] = Ax[k] + w[k]$, where $w[k]$ is zero mean white noise

According to the Lyapunov equation,

$P_{k+1} = AP_kA' + \Sigma$, $P_0 = C_0$

Let's say that matrix $A^T_{n\times n}$ is full rank, $r(A^T) = n$

(according to the attribute of positive-definite matrix,

$APkA'$ is also positive-definite if P_k is positive definite

And Because $APkA'$ and Σ are both positive-definite,

$P_{k+1} = AP_kA' + \Sigma$ is also positive definite

proof: If $P_k \geq \hat{P}_k$, we can prove that $APkA' \geq A\hat{P}kA'$ given that A^T is full rank.

$\because x'(APkA' - A\hat{P}kA')x = (A'x)'(P_k - \hat{P}_k)(A'x) \geq 0$

$\therefore APkA' \geq A\hat{P}kA'$

$\therefore APkA' + \Sigma \geq A\hat{P}kA' + \Sigma$

$\therefore P_{k+1} \geq \hat{P}_{k+1}$

So from $P_0 \geq \hat{P}_0$, we can derive $P_1 \geq \hat{P}_1$, $P_2 \geq \hat{P}_2$ ...

until $P_k \geq \hat{P}_k$

<2> First we need $E[w(k)] = 0$ to prove

$P[k+1] = E[\tilde{x}[k+1]\tilde{x}'[k+1]] = E[A\tilde{x}[k]\tilde{x}'[k]A' + A\tilde{x}[k]w'[k] + w[k]\tilde{x}'[k]A' + w[k]w'[k]]$

$= AP[k]A' + \Sigma$

Then we need the covariance matrix of noise to be semi-positive definite to prove that $P[k+1]$ is still positive definite. we know that covariance matrix is semi-definite

So Gaussianity is not required for our proof, but $\mu = 0$ and that covariance matrix to be semi-definite at least

3.

3(a) $\min \|Y-\psi\theta\|^2$, subject to constraint $A\theta=b$

① First, we solve the following classic linear regression model written as

$$Y=\psi\theta+\varepsilon \qquad \varepsilon \sim N(0, \Sigma=\sigma^2 I)$$

for this question, we proved that the classic least squares estimator is $\hat{\theta}=(\psi'\psi)^{-1}\psi'Y$

We can write the likelihood function of Y as

$$L(Y;\theta,\Sigma=\sigma^2 I)=\frac{e^{-\frac{1}{2\sigma^2}(Y-\psi\theta)'(Y-\psi\theta)}}{(2\pi)^{\frac{n}{2}}(\sigma^2)^{\frac{n}{2}}}$$

$$l=\log L=-\frac{Y'Y-2\theta'\psi'Y+\theta'\psi'\psi\theta}{2\sigma^2}-\frac{n}{2}\ln 2\pi-\frac{n}{2}\ln\sigma^2 \quad ①$$

now we have constraint $A\theta=b$, A is $m\times n$, $m<n$, θ is $n\times 1$, b is $m\times 1$

Give that there is no constraint on σ^2, we have

$$\frac{\partial l}{\partial \sigma^2}=-\frac{n}{2}\frac{1}{\sigma^2}+\frac{1}{2}\left(\frac{1}{\sigma^2}\right)^2(Y-\psi\theta^c)'(Y-\psi\theta^c)=0,$$

$\Rightarrow$ an estimator of σ^2, $\hat{\sigma}^2=\frac{1}{n}(Y-\psi\theta^c)'(Y-\psi\theta^c)$

By putting $\hat{\sigma}^2$ in equation ①, we get

$$l=\ln L=-\frac{n}{2}\ln 2\pi-\frac{n}{2}\ln\left[\frac{1}{n}(Y-\psi\theta^c)'(Y-\psi\theta^c)\right]-\frac{n}{2}$$

So now the problem is turned into a maximization problem,

$$\max_{\theta^c} l=-\frac{n}{2}\ln 2\pi-\frac{n}{2}\ln\left[\frac{1}{n}(Y-\psi\theta^c)'(Y-\psi\theta^c)\right]-\frac{n}{2}$$

$$\text{st } A\theta^c=b$$

We can write the Lagrangian function where λ' is an $m\times 1$ vector of Lagrangian multipliers

$$\text{Lagrangian}=Y'Y-2\theta^{c\prime}\psi'Y+\theta^{c\prime}\psi'\psi\theta^c-\lambda'(b-A\theta^c)$$

differentiation with respect to $\beta^{c\prime}$ and λ yields

$$\begin{cases}-2\psi'Y+2\psi'\psi\theta^c+A'\lambda=0\\ A\theta^c-b=0\end{cases}$$

$$\Rightarrow A(\psi'\psi)^{-1}(-2\psi'\psi+2\psi'\psi\theta^c+A'\lambda)=0$$

$$\Rightarrow -2A(\psi'\psi)^{-1}\psi'Y+2A\theta^c+A(\psi'\psi)^{-1}A'\lambda=0$$

Solve the above equation for λ by substituting $\hat{\theta}=(\psi'\psi)^{-1}\psi'Y$

$\Rightarrow A(\psi'\psi)^{-1}A'\lambda = 2A\hat{\beta} - 2A\theta^c$

$\Rightarrow \lambda = -2(A(\psi'\psi)^{-1}A')^{-1}(b - A\hat{\beta})$

$\because A\theta^c = b$

$-2\psi'Y + 2\psi'\psi\theta^c + A'\lambda = 0$

$\Rightarrow -2\psi'Y + 2\psi'\psi\theta^c + A'[-2(A(\psi'\psi)^{-1}A']^{-1}(b - A\hat{\beta})] = 0$

$\Rightarrow \psi'\psi\theta^c = \psi'Y + A'(A(\psi'\psi)^{-1}A')^{-1}(b - A\hat{\beta})$

$\Rightarrow \theta^c = \hat{\theta} + (\psi'\psi)^{-1}A'(A(\psi'\psi)^{-1}A')^{-1}(b - A\hat{\theta})$,

where $\hat{\theta} = (\psi'\psi)^{-1}\psi'Y$

with normally distributed errors in the model, the MLE and LSE of the constrained model are the same. So with MLE we solved this LSE problem.

② Above all is the method of MLE. Now Let's conside how to do it in LSE way.

Lagrangian is $L\theta = \|Y - \psi\theta\|^2 + \lambda'(A\theta - b)$

$\nabla L\theta = 0 \Rightarrow 2\psi'(-Y + \psi\theta) + A'\lambda \Rightarrow \theta^* = \frac{1}{2}(\psi'\psi)^{-1}(\psi'Y + A'\lambda)$

put θ^* in our constrain $A\theta - b = 0 \Rightarrow$

$\frac{1}{2}A(\psi'\psi)^{-1}(2\psi'Y + A'\lambda) - b = 0$

$\Rightarrow \lambda = +2[A(\psi'\psi)^{-1}A']^{-1}[A(\psi'\psi)^{-1}\psi'Y - b]$

put λ in $\theta^* = \frac{1}{2}(\psi'\psi)^{-1}(2\psi'Y + A\lambda)$

we have $\theta^* = \frac{1}{2}(\psi'\psi)^{-1}\{2\psi'Y + A'(+2)(A(\psi'\psi)^{-1}A')^{-1}(A(\psi'\psi)^{-1}\psi'Y - b)\}$.

$= \hat{\theta} + (\psi'\psi)^{-1}A'(A(\psi'\psi)^{-1}A')^{-1}(b - A\hat{\theta})$

where $\hat{\theta} = (\psi'\psi)^{-1}\psi'Y$

We can see MLE and LSE are the same.

3 (b) $\min F(\theta) = \|Y - \psi\theta\|^2$, subject to $A\theta = b$

Lagrangian

$L(\theta, \lambda) = \|Y - \psi\theta\|^2 + \lambda'(A\theta - b)$

According to primal dual algorithm, we have

$\theta^{(n+1)} = \theta^{(n)} - \epsilon_n \nabla_\theta L(\theta^{(n)}, \lambda^{(n)}) = \theta^{(n)} + \epsilon_n (-2X\psi'(Y - \psi\theta) + A'\lambda)$

$\lambda^{(n+1)} = \lambda^{(n)} + \epsilon_n \nabla_\lambda L(\theta^{(n)}, \lambda^{(n)})$

$= \lambda^{(n)} + \epsilon_n (A\theta - b)$

we set $\theta = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]'$

$A = [1, 1, 0, 0, 0, 0, 0, 0, 0, 0]$

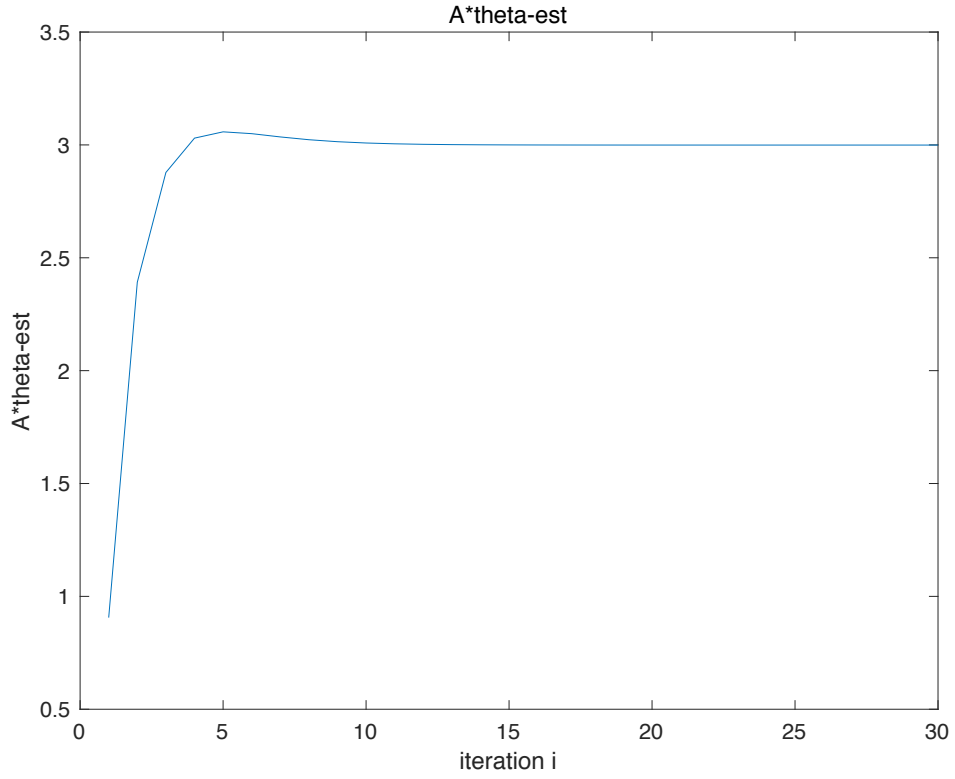
$b = A\theta = 3$

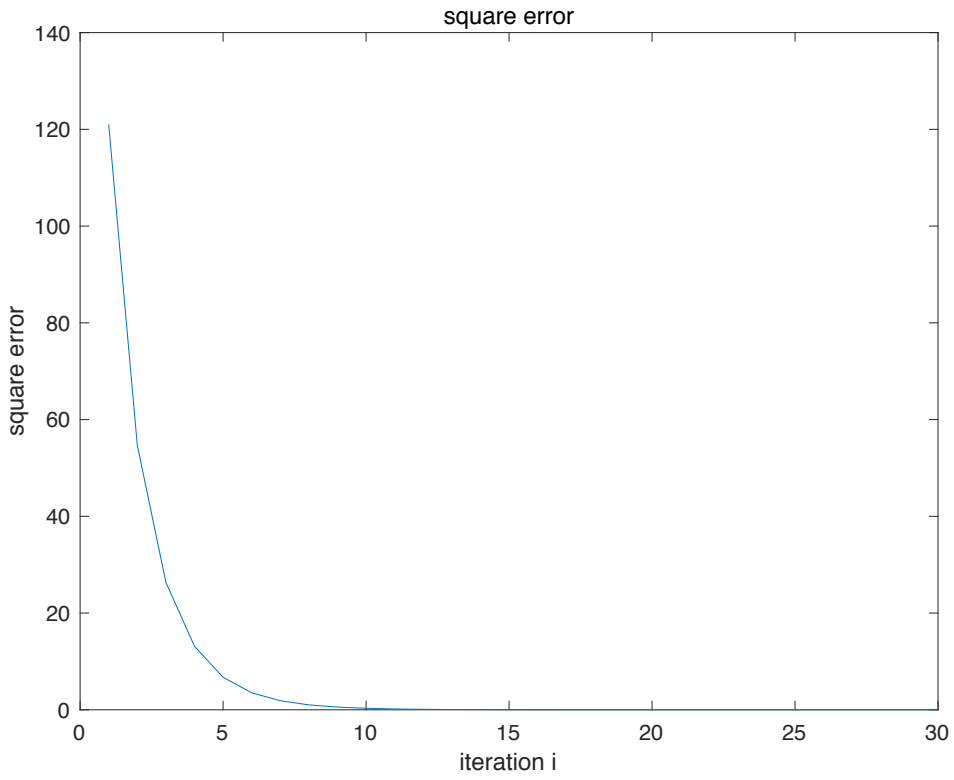
for convenience, suppose there is no noise, so $Y = \psi * \theta$

We set step $\epsilon_n = 0.01 \times 0.99^i$

we plot the figure of changes of norm $\|Y - \psi\theta\|$ and $A\theta\_est$

We can see that θ converges very fast (with in 20 points)




square error
140
120
100
80
60
40
20
0
square error
0
5
10
15
20
25
30
iteration i

4

4 (a) When $u=1,2$, we let $c(u,i)=\frac{1}{i}$

We set P_1, P_2 in the code to achieve that $\sum_{j\geq l} P_{ij}(u) < \sum_{j\geq l} P_{i+1,j}(u)$ for $l=1,2,3,\cdots 20$

By changing the P, we check the vector $V_k(i)$ and we can find that $V_k(i)$ is a decreasing function.

(b) when $\rho=0.9$, $V_{opt}(1)=9.3329$

when $\rho=0.7$, $V_{opt}(1)=3.2119$

when $\rho=0.5$, —— $=1.9157$

—— 0.3 —— 1.4083

—— 0.1 —— 1.106

So when ρ increases, the optimal cumulative cost value will increase

(c) Yes, the average is close to the optimal cumulative cost from matlab we can see that our $V(i,k)$ converges very fast.

We use Piso, value iteration also to write the code.

$V_k(i)=\min_{u\in N} Q_k(i,u)$, $\mu^*_k(i)=\arg\min_{u\in\mu} Q_k(i,u)$

$Q_k(i,\mu)=c(i,u)+\rho\sum_j P_{ij}(u)V_{k-1}(j)$.

(a) For a fixed θ, we generate x by a Gaussian pdf with mean θ and variance. And we put x through our black box to get output $C(x)$. We do this N times, and get mean of all the output $C(x)$. In this way we can get $E_{p_\theta}\{C(x)\}$

SPSA Algo

(b) ① Simulate the p dimensional vector d_n with random element

$$d_n(i) = \begin{cases} -1 & \text{with } p=0.5 \\ +1 & \text{with } p=0.5 \end{cases}$$

② Evaluate sample costs $\hat{C}_n(\theta_n + \Delta_n d_n)$ and $\hat{C}_n(\theta_n - \Delta_n d_n)$

③ Compute gradient estimate along direction d_n:

$$\hat{\nabla} C(\theta_n) = \frac{\hat{C}_n(\theta_n + \Delta_n d_n) - \hat{C}_n(\theta_n - \Delta_n d_n)}{2\Delta_n} d_n,$$

where $\Delta_n = \frac{\Delta}{(n+1)^r}$, $0.5 \le r \le 1$, $\Delta > 0$

④ $\hat{\nabla} C(\theta_n)$ can be considered as a estimation for $\frac{d}{d\theta} E_{p_\theta}\{C(x)\}$

(c) According to the stochastic gradient algorithm, in order to find θ that minimize $E_{p_\theta}\{C(x)\}$ we can use the following:

$$\theta_{n+1} = \theta_n - \epsilon_n \hat{\nabla} C(\theta_n), \quad \epsilon_n = \frac{\epsilon}{(n+1+S)^\beta},$$

$$S > 0, \ \epsilon > 0, \ 0 < \beta \le 1$$

However, after applying SPSA Algo I found it hard to converge So I used another method to estimate the $\hat{\nabla}(C(\theta_n))$.

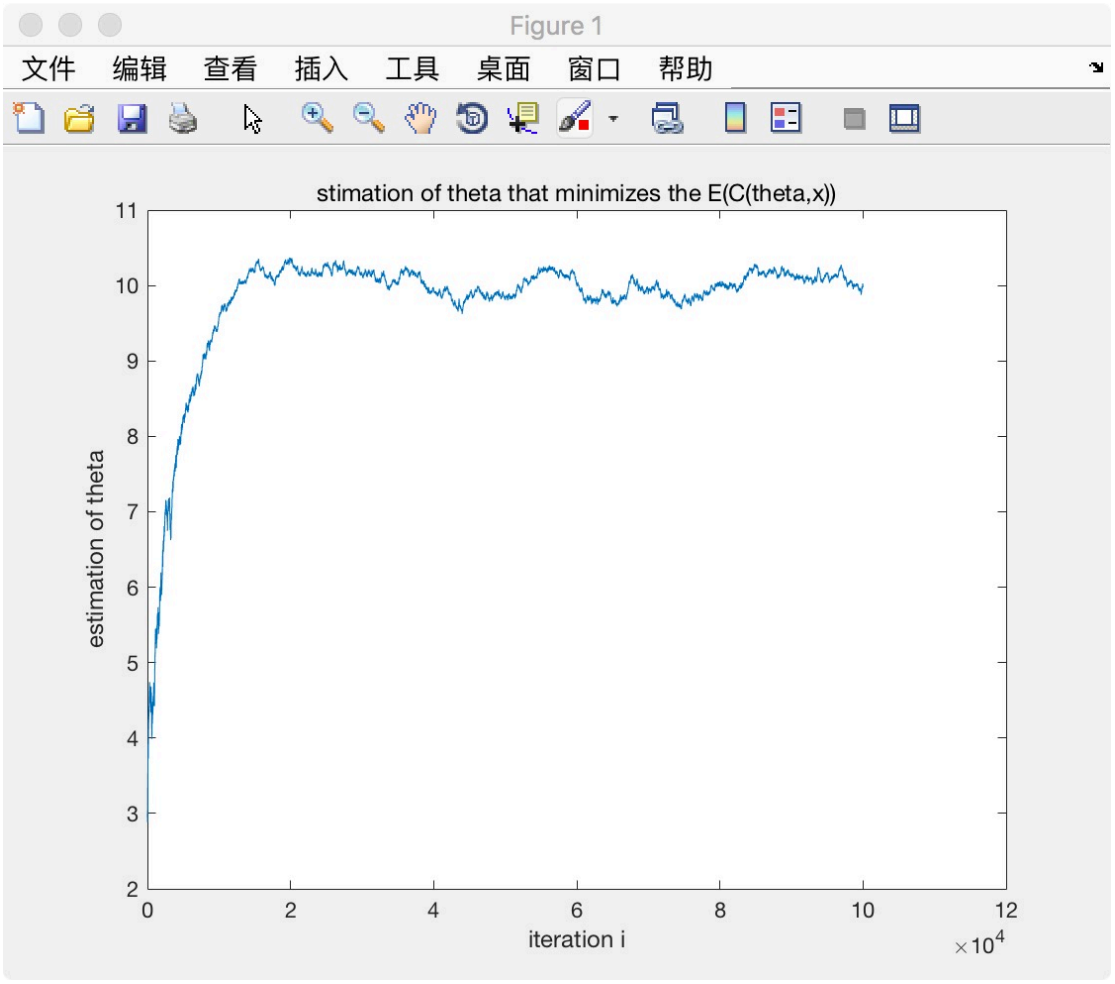
① generate 10 $C_i(\theta + 0.01)$ and 10 $C_i(\theta - 0.01)$, $i = 1, 2, \ldots, 10$

$$\hat{\nabla} C(\theta_n) = \frac{\sum_{i=1}^{10} C_i(\theta + 0.01) - \sum_{i=1}^{10} C_i(\theta - 0.01)}{0.02 \times 10}$$

② Then we use $\theta_{n+1} = \theta_n - \epsilon_n \hat{\nabla} C(\theta_n)$ to do the estimate

We let $C(x,\theta) = (X-10)^2$, so ideally when $\theta = 0$, $E[C(X,\theta)]$ will be the smallest.

As in the image we can see that θ converges to 10 after enough iterations.



First of all, I consider the concept of Markov chain. Markov chain helps us understand better of the world in area of probability. We learned the definition of Markov chain and how to calculate it with transition matrix and distribution vector. We then further learned how to simulate a Markov chain. It is useful when we want to confirm some theory of Markov chain because we can just simulate it and run the result to check if it matched the ideal value in the theory. We also learned how to calculate the entropy of the Markov chain and this helps us learn what kind of system contains most information. One example we learned of Markov chain that is applied in the real world application is Google Page Rank Algorithm. The whole key idea is that webpage importance = # pages linked to it. So by using the Markov chain to show the link connection between each webpage we can simulate the probability one webpage will get linked to and thus know the importance of each webpage. Another example is to use Markov chain to build the queuing model. This helps us analyze the condition of each queue and the total cost of people in all the lines waiting to be served. And thus it is the very basic of Markov Decision Processes to help us learn how to arrange the service rule to minimize the total cost of people waiting to get served. By searching the internet, we will find even more interesting stuff. We can use knowledge of Markov chain to predict the price of the stock, the price of the land , the GDP of an area and even more interesting stuff.

Secondly, what comes to my mind is the concept of least squares estimation. It's very basic solution for the overdetermined solution is easy to derive. I have already learned this very basic thing before but not in this detail. I learn that persistent exciting inputs are necessary for us to get X' X to be reversible and thus least squares' columns of X are linearly independent so that all the modes of the system are excited enough. Then we use the geometric method to illustrate the help understand more clearly of least squares estimation. Then we know that Matlab will not just directly calculate our function of least squares but in a cleverer way which use the conception of SVD (Singular Value Decomposition). It helps the Matlab to better solve the big matrix computation problem and decrease the computation time. Then we get in touch with the concept of batch-processing algorithms (off line) and recursive algorithms (on line). The online algorithm called recursive least squares is widely used in all areas of internet. It Updated model

parameter estimate theta_k with each new data point. Online algorithm is critical for social network because the users generate the data all the time and you will never have time to wait for them to generate the whole bunch of data for you and you start to work. Similarly, recursive least squares can be applied in many areas. For example, it can be applied in the airplane control system. It will update the data all the time from the plane control system and tell the working condition on-line by predicting the flying parameters. It' s used to design the plane before engineers build it. We obviously can not afford the cost to try in real whether a plane can work properly or not. So by recursive least squares we can simulate the flying system and better the designing. Least squares algorithm is also widely used in the economics area just like Markov chain. Business man can use LS algorithm to predict next year' s sale situation and accordingly set the plan of production to avoid over-produced items.

Last but not the least, what comes to my mind is the concept of stochastic gradient algorithm. To do the stochastic gradient algorithm we need to know how to do the gradient estimation. We learn two methods to do this, Kiefer Wolfowitz algorithm and SPSA Algorithm. Then we learn the most famous least mean square algorithm. It's derived from the general stochastic gradient algorithm and very powerful. Finally we changed the augmented Lagrangian algorithm into stochastic augmented Lagrangian algorithm which involves the gradient estmation for the problem of constrained stochastic optimization. Least mean square is widely used in auto-adaptive filters. So we can use LMS algorithm to develop the algorithm to filter out the noise and this can be used to build noise cancelling headphone like SONY MDR-1000X and BOSE QR35 series. After searching for information on the internet I find that LMS algorithm does not necessarily need the step to be fixed. We can build the algorithm to change the step to make it suitable for satellite signal processing since satellite signal is always filled by a lot of noise. For SPSA algorithm, government of China has already utilized it in the social transportation shuttle bus arrangement. It will consider many factors like traffic situation, arrival time of each bus, the number of passengers waiting at the station to arrange the bus departure from the center and thus successfully increase the efficiency of the whole social traffic system and improve the experience of passengers.