**Solution 1: Online Updates**

Suppose $z_1, \dots, z_t \in \mathbb{R}^d$ are the environmental data points seen until time $t \in \mathbb{N}$.

(a) Provide an update formula for the empirical mean of the data points for any time instance $s = 1, \dots, t$ in form of a function $u : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{N} \to \mathbb{R}^d$, such that

$$\bar{z}_s = u(\bar{z}_{s-1}, z_s, s),$$

holds. Here, $\bar{z}_s = \frac{1}{s} \sum_{j=1}^s z_j$ denotes the empirical mean at time $s$ and we have the convention that $\sum_{j=1}^s z_j = 0$, if $s = 0$.

**Solution:**

Note that we can write

$$\begin{aligned}
\bar{z}_s &= \frac{1}{s} \sum_{j=1}^s z_j \\
&= \frac{1}{s} \left( \sum_{j=1}^{s-1} z_j + z_s \right) \\
&= \frac{1}{s} \left( \frac{s-1}{s-1} \cdot \sum_{j=1}^{s-1} z_j + z_s \right) \\
&= \frac{1}{s} \left( (s-1)\bar{z}_{s-1} + z_s \right).
\end{aligned}$$

Thus, the function $u(a, b, c) := \frac{1}{c}((c-1)a + b)$ is such that $\bar{z}_s = u(\bar{z}_{s-1}, z_s, s)$ holds for any time step $s = 1, \dots, t$.

(b) Provide an update formula for the empirical total variance of the data points for any time instance $s = 1, \dots, t$ in form of a function $u : \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{N} \to \mathbb{R}$, such that

$$v_s = u(\overline{z_{s-1}^2}, \bar{z}_{s-1}, z_s, s)$$

with $v_s = \frac{1}{s} \sum_{j=1}^s (z_j - \bar{z}_s)^\top (z_j - \bar{z}_s)$ holds. Here, $\overline{z_s^2} = \frac{1}{s} \sum_{j=1}^s z_j^\top z_j$ denotes the empirical mean of the inner products of the data points at time $s$.

**Solution:**

Note that

$$\begin{aligned}
v_s &= \frac{1}{s} \sum_{j=1}^s (z_j - \bar{z}_s)^\top (z_j - \bar{z}_s) \\
&= \frac{1}{s} \sum_{j=1}^s z_j^\top z_j - 2\bar{z}_s^\top z_j + \bar{z}_s^\top \bar{z}_s \\
&= \frac{1}{s} \sum_{j=1}^s z_j^\top z_j - \frac{2}{s} \bar{z}_s^\top \sum_{j=1}^s z_j + \bar{z}_s^\top \bar{z}_s \\
&= \overline{z_s^2} - \bar{z}_s^\top \bar{z}_s.
\end{aligned}$$

Since $\overline{z_s^2}$ is essentially an empirical average (of inner products), we can use almost the same update function as for the empirical mean in (a):

$$\overline{z_s^2} = u_1(\overline{z_{s-1}^2}, z_s, s),$$

where $u_1(a,b,c) = \frac{1}{c}((c-1)a + b^\top b)$. For $\bar{z}_s^\top \bar{z}_s$ we can use the update function for the mean by computing the inner product of the two update functions:

$$\bar{z}_s^\top \bar{z}_s = (u_2(\bar{z}_{s-1}, z_s, s))^\top u_2(\bar{z}_{s-1}, z_s, s),$$

where $u_2(a,b,c) = \frac{1}{c}((c-1)a + b)$. Combining everything together yields

$$v_s = \overline{z_s^2} - \bar{z}_s^\top \bar{z}_s = u_1(\overline{z_{s-1}^2}, z_s, s) - (u_2(\bar{z}_{s-1}, z_s, s))^\top u_2(\bar{z}_{s-1}, z_s, s) =: u(\overline{z_{s-1}^2}, \bar{z}_{s-1}, z_s, s),$$

where $u(a,b,c,s) = u_1(a,c,s) - (u_2(b,c,s))^\top u_2(b,c,s)$.

(c) Explain the benefits of having such update formulas for a particular statistic in the online learning framework.

**Solution:**

The dynamic aspects of online learning problems necessitate a mechanism for rapid action determination. As a consequence, it is crucial to have update formulas for various statistics (especially omnipresent statistics as the empirical mean and variance), which do **not** use the entire data to compute the current (time-dependent) value of the statistic if new data is available, but only a sufficiently small portion of previous data.

**Solution 2: Practical Performance of FTL and FTRL**

(a) Consider an online quadratic optimization problem with action space $\mathcal{A} = [-1,1]^d$, environment data space $\mathcal{Z} = [-1,1]^d$ and the loss function given by $L(a,z) = \frac{1}{2}||a-z||_2^2$. Furthermore, let $T = 10000$ be the considered time horizon. Assume that the environmental data is generated uniformly at random in each time step $t \in \{1,\dots,T\}$. Compute the cumulative regret of FTL and of FTRL instantiated with the squared L2-norm regularization and the optimal choice for the regularization magnitude for any time step $t = 1,\dots,T$.

Repeat this procedure 100 times and compute the empirical average of the resulting cumulative regrets in each time step $t = 1,\dots,T$. Note that this results in a curve with support points $(t, \bar{R}_t^{\texttt{Algo}})_{t=1,\dots,T}$, where $\bar{R}_t^{\texttt{Algo}}$ is the average cumulative regret (over the 100 repetitions) of algorithm $\texttt{Algo} \in \{\text{FTL}, \text{FTRL}\}$ till time step $t$. Plot this mean cumulative regret curve together with the theoretical upper bound for the cumulative regret of the FTRL algorithm in this case into one chart. Include also the empirical standard error of the mean cumulative regret, i.e., include the points $(t, \bar{R}_t^{\texttt{Algo}} \pm \hat{\sigma}(R_t^{\texttt{Algo}}))_{t=1,\dots,T}$, where $\hat{\sigma}(R_t^{\texttt{Algo}})$ is the empirical standard deviation of the cumulative regret (over the 100 repetitions) of algorithm $\texttt{Algo} \in \{\text{FTL}, \text{FTRL}\}$ till time step $t$. How does the mean cumulative regret curve of FTRL vary with respect to the regularization magnitude? Illustrate this also by means of a chart.

*Hint:* For $d$ you can, of course, consider different settings.

(b) Repeat (a), but this time consider an online linear optimization problem with the same action space, the same environment data space, but with the loss function given by $L(a,z) = a^\top z$. Comment on your findings.

**Solution:**

Note that

- the cumulative quadratic loss for a fixed action $a$ in time step $t$ is

$$\sum_{s=1}^t \frac{1}{2}||a - z_s||_2^2 = \sum_{s=1}^t \left(\frac{1}{2}a^\top a - 2a^\top z_s + z_s^\top z_s\right) = \frac{t}{2}a^\top a - a^\top \sum_{s=1}^t z_s + \frac{1}{2}\sum_{s=1}^t z_s^\top z_s$$

- if we use the $l_2$-norm regularization, i.e.,

$$\psi(a) = \frac{1}{2\eta}||a||_2^2$$

for the FTRL algorithm, we have that

$$a_{t+1}^{\text{FTRL}} = \frac{1}{\frac{1}{\eta} + t}\sum_{s=1}^t z_s = \frac{1}{\frac{1}{\eta} + t}\left((t-1)\overline{z_{t-1}} + z_t\right)$$

```r
set.seed(123)

################################################################################
################################################################################
### First part: Online quadratic optimization problem
################################################################################
################################################################################

# define the quadratic loss function

# a is the action
# z is the environmental data

lquad <- function(a, z){
  return (0.5*sum((a-z)^2))
}

# Computing the cumulative quadratic loss of a particular action
# -> necessary to compute the cumulative loss of the best action in hindsight

# a is the action
# z_t_tilde is the sum over all environmental data till time t
# z_t_tilde_square is the sum over the inner products of all environmental data
# z_t is the new data in t
# t is the current time step

cum_quad_loss <-function(a, z_t_tilde, z_t_tilde_square, z_t,t){
  return ( t/2*sum(a*a) - a%*% z_t_tilde + 1/2*(z_t_tilde_square))
}

# define the FTL algorithm for quadratic optimzation
# -> use the update formula for empirical average

# z_t_1 is the emp. average of the data over the time steps before t
# z_t is the new data in t
# t is the current time step

FTL_quad <- function(z_t_1, z_t, t){
  return (1/t*((t-1)*z_t_1+z_t))
}

# define the FTRL algorithm for quadratic optimzation
# -> use the update formula for empirical average

# z_t_1 is the emp. average of the data over the time steps before t
# z_t is the new data in t
# t is the current time step
# eta is the regularization parameter in the l_2 norm regularization

FTRL_quad <- function(z_t_1, z_t, t, eta){
  return (1/(1/eta+t)*((t-1)*z_t_1+z_t))
}



################################################################################
# simulation framework for online quadratic optimization


# time horizon
T        = 10000
# repition number of simulations
rep_num = 20
```

```r
# dimension
d       = 10

# to store the averaged regret in the end
regret_FTL_all    = matrix(rep(0,rep_num*T),nrow=rep_num)
regret_FTRL_all   = matrix(rep(0,rep_num*T),nrow=rep_num)

# to store the euclidean norm of the best actions in hindsight
best_actions_norm = rep(0,rep_num)

# we have not derived an optimal choice for eta in this scenario,
# thus we set it simply to 1
eta = 1

for (i in 1:rep_num){

  a_t_FTL           = rep(0,d) # action of FTL(first time is set to zero)
  a_t_FTRL          = rep(0,d) # action of FTRL(first time is set to zero)
  z_t_mean          = rep(0,d) # empirical mean of the environment data at
                               # time step t
  z_t_tilde         = rep(0,d) # the sum over all environmental data
  z_t_tilde_square = 0         # the sum over the inner products of all
                               # environmental data
  regret_FTL        = rep(0,T) # counting the regret of FTL over the time horizon
  regret_FTRL       = rep(0,T) # counting the regret of FTRL over the time horizon
  loss_best_action = rep(0,T) # the cumulative loss of the best action in t
  loss_FTL          = rep(0,T) # the cumulative loss of FTL
  loss_FTRL         = rep(0,T) # the cumulative loss of FTRL

    for (t in 1:T){

      # generate environment data uniformly at random
      z_t                   = runif(d,-1,1)
      # update the sum over all data
      z_t_tilde             = z_t_tilde + z_t
      # update the sum over the inner products of the data
      z_t_tilde_square     = z_t_tilde_square + sum(z_t*z_t)
      # update the mean
      z_t_mean              = FTL_quad(z_t_mean, z_t, t)
      # FTL uses exactly the mean
      # compute the cumulative loss of the best action in hindsight,
      #which is the emp. mean in t
      loss_best_action[t] = cum_quad_loss(z_t_mean, z_t_tilde, z_t_tilde_square,
                                          z_t, t)
      # compute the cumulative loss of FTL resp. FTRL
      loss_FTL[t]           = loss_FTL[max(1,t-1)]   + lquad(a_t_FTL,z_t)
      loss_FTRL[t]          = loss_FTRL[max(1,t-1)] + lquad(a_t_FTRL,z_t)
      # compute the regret of FTL resp. FTRL for the current time step
      regret_FTL[t]         = loss_FTL[t]    - loss_best_action[t]
      regret_FTRL[t]        = loss_FTRL[t]   - loss_best_action[t]

      # update the actions of FTL and FTRL
      a_t_FTRL              = FTRL_quad(a_t_FTL, z_t, t, eta)
      #  a_t_FTL is the emp. mean over the data till t-1
      a_t_FTL               = z_t_mean
      #  a_t_FTL becomes the emp. mean over the data till t

    }

  best_actions_norm[i]     = sum(z_t_mean*z_t_mean)
  regret_FTL_all[i,]       = regret_FTL
  regret_FTRL_all[i,]      = regret_FTRL

}
```

```
# plot everything

legend_vec =c("FTL","Upper bound","FTRL")

plot(1:T,apply(regret_FTL_all,2,mean),type="l",
     ylim=c(0,max(4*sqrt(d)*((log(1:T))+1))),ylab="R_t",xlab="t")
lines(1:T,apply(regret_FTL_all,2,mean)+apply(regret_FTL_all,2,sd))
lines(1:T,apply(regret_FTL_all,2,mean)-apply(regret_FTL_all,2,sd))
lines(1:T,apply(regret_FTRL_all,2,mean),type="l",col=3)
lines(1:T,apply(regret_FTRL_all,2,mean)+apply(regret_FTRL_all,2,sd),type="l",
      col=3)
lines(1:T,apply(regret_FTRL_all,2,mean)-apply(regret_FTRL_all,2,sd),type="l",
      col=3)
lines(1:T,4*sqrt(d)*((log(1:T))+1),col=2)
legend("right",legend=legend_vec,col=1:3,lty=rep(1,3),cex=1.5)

# plot the euclidean norm of the best action in hindsight
plot(1:rep_num,best_actions_norm,xlab="rep_num",
     ylab="Euclidean norm best action")

##############################################################################
# simulation framework for online quadratic optimization with
# varying reg. parameter eta for FTRL


# time horizon
T        = 10000
# repition number of simulations
rep_num = 20
# dimension
d        = 10

eta_start = 0.2
eta_end   = 5
eta_grid  = seq(eta_start,eta_end,l=8)

par(mfrow=c(4,2))

for (eta in eta_grid) {

  # to store the averaged regret in the end
  regret_FTL_all     = matrix(rep(0,rep_num*T),nrow=rep_num)
  regret_FTRL_all    = matrix(rep(0,rep_num*T),nrow=rep_num)

  # to store the euclidean norm of the best actions in hindsight
  best_actions_norm = rep(0,rep_num)



  for (i in 1:rep_num){

    a_t_FTL           = rep(0,d) # action of FTL(first time is set to zero)
    a_t_FTRL          = rep(0,d) # action of FTRL(first time is set to zero)
    z_t_mean          = rep(0,d) # the empirical mean of the environment data
                                 # at time step t
    z_t_tilde         = rep(0,d) # the sum over all environmental data
    z_t_tilde_square = 0         # the sum over the inner products of all
                                 # environmental data
    regret_FTL        = rep(0,T) # counting the regret of FTL over the time horizon
    regret_FTRL       = rep(0,T) # counting the regret of FTRL over the time horizon
    loss_best_action = rep(0,T) # the cumulative loss of the best action in t
    loss_FTL          = rep(0,T) # the cumulative loss of FTL
```

```
    loss_FTRL         = rep(0,T) # the cumulative loss of FTRL

    for (t in 1:T){

      # generate environment data uniformly at random
      z_t                 = runif(d,-1,1)
      # update the sum over all data
      z_t_tilde           = z_t_tilde + z_t
      # update the sum over the inner products of the data
      z_t_tilde_square    = z_t_tilde_square + sum(z_t*z_t)
      # update the mean
      z_t_mean            = FTL_quad(z_t_mean, z_t, t)
      # FTL uses exactly the mean
      # compute the cumulative loss of the best action in hindsight,
      # which is the emp. mean in t
      loss_best_action[t] = cum_quad_loss(z_t_mean, z_t_tilde, z_t_tilde_square,
                                          z_t, t)
      # compute the cumulative loss of FTL resp. FTRL
      loss_FTL[t]         = loss_FTL[max(1,t-1)]  + lquad(a_t_FTL,z_t)
      loss_FTRL[t]        = loss_FTRL[max(1,t-1)] + lquad(a_t_FTRL,z_t)
      # compute the regret of FTL resp. FTRL for the current time step
      regret_FTL[t]       = loss_FTL[t]   - loss_best_action[t]
      regret_FTRL[t]      = loss_FTRL[t]  - loss_best_action[t]

      # update the actions of FTL and FTRL
      a_t_FTRL            = FTRL_quad(a_t_FTL, z_t, t, eta)
      #  a_t_FTL is the emp. mean over the data till t-1
      a_t_FTL             = z_t_mean
      #  a_t_FTL becomes the emp. mean over the data till t

    }

    best_actions_norm[i]     = sum(z_t_mean*z_t_mean)
    regret_FTL_all[i,]       = regret_FTL
    regret_FTRL_all[i,]      = regret_FTRL

  }


  # plot everything

  plot(1:T,apply(regret_FTL_all,2,mean),type="l",ylab="R_t",xlab="t",
       main=paste("eta = ", eta))
  lines(1:T,apply(regret_FTL_all,2,mean)+apply(regret_FTL_all,2,sd))
  lines(1:T,apply(regret_FTL_all,2,mean)-apply(regret_FTL_all,2,sd))
  lines(1:T,apply(regret_FTRL_all,2,mean),type="l",col=3)
  lines(1:T,apply(regret_FTRL_all,2,mean)+apply(regret_FTRL_all,2,sd),type="l",
        col=3)
  lines(1:T,apply(regret_FTRL_all,2,mean)-apply(regret_FTRL_all,2,sd),type="l",
        col=3)
  lines(1:T,4*sqrt(d)*((log(1:T))+1),col=2)
  #legend("right",legend=legend_vec,col=1:3,lty=rep(1,3),cex=1.5)

}

################################################################################
################################################################################
### Second part: Online linear optimization problem
################################################################################
################################################################################



# define the linear loss function
```

```r
# a is the action
# z is the environmental data

llinear <- function(a, z){
  return (sum(a*z))
}


# Computing the cumulative linear loss of a particular action
# -> necessary to compute the cumulative loss of the best action in hindsight

# a is the action
# z_t_tilde is the sum over all environmental data till time t

cum_linear_loss <- function(a, z_t_tilde){
  return (sum(a *z_t_tilde) )
}


# define the FTL algorithm for linear optimzation
# -> use the insights of the example in the lecture
# -> since A=[-1,1]^d FTL returns the negative signs in each component of
#    the z_t_tilde vector

# z_t_tilde is the sum over all environmental data till time t

FTL_lin <- function(z_t_tilde){
  return (-sign(z_t_tilde))
}

# define the FTRL algorithm for linear optimzation
# -> use the update formula in the lecture

# a_prev is the previous action of FTRL
# z_t is the new data in t
# eta is the regularization parameter in the l_2 norm regularization

FTRL_lin <- function(a_prev, z_t,  eta){
  # since A=[-1,1]^d it could happen that we are running out of the action space
  #therefore we need to project back to A
  return (pmin(pmax(a_prev - eta*z_t,-rep(1,d)),rep(1,d)))
}



##############################################################################
# simulation framework for online linear optimization


# time horizon
T       = 10000
# repition number of simulations
rep_num = 20
# dimension
d       = 10

# to store the averaged regret in the end
regret_FTL_all     = matrix(rep(0,rep_num*T),nrow=rep_num)
regret_FTRL_all    = matrix(rep(0,rep_num*T),nrow=rep_num)


par(mfrow=c(1,1))
```

```r
# optimal choice of eta
eta = 1/(sqrt( 2 * T ))

for (i in 1:rep_num){

  a_t_FTL          = rep(0,d) # action of FTL(first time is set to zero)
  a_t_FTRL         = rep(0,d) # action of FTRL(first time is set to zero)
  z_t_mean         = rep(0,d) # the empirical mean of the environment data
                              # at time step t
  z_t_tilde        = rep(0,d) # the sum over all environmental data
  z_t_tilde_square = 0        # the sum over the inner products of all
                              # environmental data
  regret_FTL       = rep(0,T) # counting the regret of FTL over the time horizon
  regret_FTRL      = rep(0,T) # counting the regret of FTRL over the time horizon
  loss_best_action = rep(0,T) # the cumulative loss of the best action in t
  loss_FTL         = rep(0,T) # the cumulative loss of FTL
  loss_FTRL        = rep(0,T) # the cumulative loss of FTRL

  for (t in 1:T){

    # generate environment data uniformly at random
    z_t                 = runif(d,-1,1)
    # update the sum over all data
    z_t_tilde           = z_t_tilde + z_t
    # compute the cumulative loss of the best action in hindsight,
    # which is the vector with the negative signs of each component of z_t_tilde
    loss_best_action[t] = cum_linear_loss(-sign(z_t_tilde), z_t_tilde)
    # compute the cumulative loss of FTL resp. FTRL
    loss_FTL[t]         = loss_FTL[max(1,t-1)]  + llinear(a_t_FTL,z_t)
    loss_FTRL[t]        = loss_FTRL[max(1,t-1)] + llinear(a_t_FTRL,z_t)
    # compute the regret of FTL resp. FTRL for the current time step
    regret_FTL[t]       = loss_FTL[t]   - loss_best_action[t]
    regret_FTRL[t]      = loss_FTRL[t]  - loss_best_action[t]

    # update the actions of FTL and FTRL
    a_t_FTRL            = FTRL_lin(a_t_FTL, z_t,  eta)
    a_t_FTL             = FTL_lin(z_t_tilde)

  }

  regret_FTL_all[i,]       = regret_FTL
  regret_FTRL_all[i,]      = regret_FTRL

}


# plot everything

legend_vec =c("FTL","Upper bound","FTRL")

plot(1:T,apply(regret_FTL_all,2,mean),type="l",
     ylim=c(0,max(sqrt(2)*d*((sqrt(1:T))))),ylab="R_t",xlab="t")
lines(1:T,apply(regret_FTL_all,2,mean)+apply(regret_FTL_all,2,sd))
lines(1:T,apply(regret_FTL_all,2,mean)-apply(regret_FTL_all,2,sd))
lines(1:T,apply(regret_FTRL_all,2,mean),type="l",col=3)
lines(1:T,apply(regret_FTRL_all,2,mean)+apply(regret_FTRL_all,2,sd),type="l",
      col=3)
lines(1:T,apply(regret_FTRL_all,2,mean)-apply(regret_FTRL_all,2,sd),type="l",
      col=3)
lines(1:T,sqrt(2)*d*((sqrt(1:T))),col=2)
legend("right",legend=legend_vec,col=1:3,lty=rep(1,3),cex=1.5)

###############################################################################
# simulation framework for online quadratic optimization with
```

```r
# varying reg. parameter eta for FTRL


# time horizon
T        = 10000
# repition number of simulations
rep_num = 20
# dimension
d        = 10

eta_start = min( 1/(sqrt( 2 * T )),  0.000001)
eta_end   = 1/(sqrt( 2 * T )) + 3
eta_grid  = seq(eta_start,eta_end,l=8)

par(mfrow=c(4,2))

for (eta in eta_grid) {

  # to store the averaged regret in the end
  regret_FTL_all     = matrix(rep(0,rep_num*T),nrow=rep_num)
  regret_FTRL_all    = matrix(rep(0,rep_num*T),nrow=rep_num)


  for (i in 1:rep_num){

    a_t_FTL           = rep(0,d) # action of FTL(first time is set to zero)
    a_t_FTRL          = rep(0,d) # action of FTRL(first time is set to zero)
    z_t_mean          = rep(0,d) # the empirical mean of the environment data
                                 # at time step t
    z_t_tilde         = rep(0,d) # the sum over all environmental data
    z_t_tilde_square = 0         # the sum over the inner products of all
                                 # environmental data
    regret_FTL        = rep(0,T) # counting the regret of FTL over the time horizon
    regret_FTRL       = rep(0,T) # counting the regret of FTRL over the time horizon
    loss_best_action = rep(0,T) # the cumulative loss of the best action in t
    loss_FTL          = rep(0,T) # the cumulative loss of FTL
    loss_FTRL         = rep(0,T) # the cumulative loss of FTRL

    for (t in 1:T){

      # generate environment data uniformly at random
      z_t                   = runif(d,-1,1)
      # update the sum over all data
      z_t_tilde             = z_t_tilde + z_t
      # compute the cumulative loss of the best action in hindsight,
      # which is the vector with the negative signs of each component of z_t_tilde
      loss_best_action[t] = cum_linear_loss(-sign(z_t_tilde), z_t_tilde)
      # compute the cumulative loss of FTL resp. FTRL
      loss_FTL[t]           = loss_FTL[max(1,t-1)]  + llinear(a_t_FTL,z_t)
      loss_FTRL[t]          = loss_FTRL[max(1,t-1)] + llinear(a_t_FTRL,z_t)
      # compute the regret of FTL resp. FTRL for the current time step
      regret_FTL[t]         = loss_FTL[t]   - loss_best_action[t]
      regret_FTRL[t]        = loss_FTRL[t]  - loss_best_action[t]

      # update the actions of FTL and FTRL
      a_t_FTRL              = FTRL_lin(a_t_FTL, z_t,  eta)
      a_t_FTL               = FTL_lin(z_t_tilde)

    }

    regret_FTL_all[i,]       = regret_FTL
    regret_FTRL_all[i,]      = regret_FTRL

  }
```

```r
  # plot everything

  plot(1:T,apply(regret_FTL_all,2,mean),type="l",ylab="R_t",xlab="t",
       main=paste("eta = ", round(eta,4)))
  lines(1:T,apply(regret_FTL_all,2,mean)+apply(regret_FTL_all,2,sd))
  lines(1:T,apply(regret_FTL_all,2,mean)-apply(regret_FTL_all,2,sd))
  lines(1:T,apply(regret_FTRL_all,2,mean),type="l",col=3)
  lines(1:T,apply(regret_FTRL_all,2,mean)+apply(regret_FTRL_all,2,sd),type="l",
        col=3)
  lines(1:T,apply(regret_FTRL_all,2,mean)-apply(regret_FTRL_all,2,sd),type="l",
        col=3)
  lines(1:T,sqrt(2)*d*((sqrt(1:T))),col=2)

}
```