Xun Cao GTID: 903237377

(a1)

Calculations are based on following relation:

$S_i = [S_{i-1} - Q_{i-1}] + W_{i-1} - D_{i-1}$

$\mathrm{Q} = 135.147 \times [\mathrm{H} - 1131.388]^{1.679}$

$\mathrm{H} = 733.73 - 0.00155\ \mathrm{S} + 50.63\ \mathrm{lnS}$

Substitute H into Q, we can get:

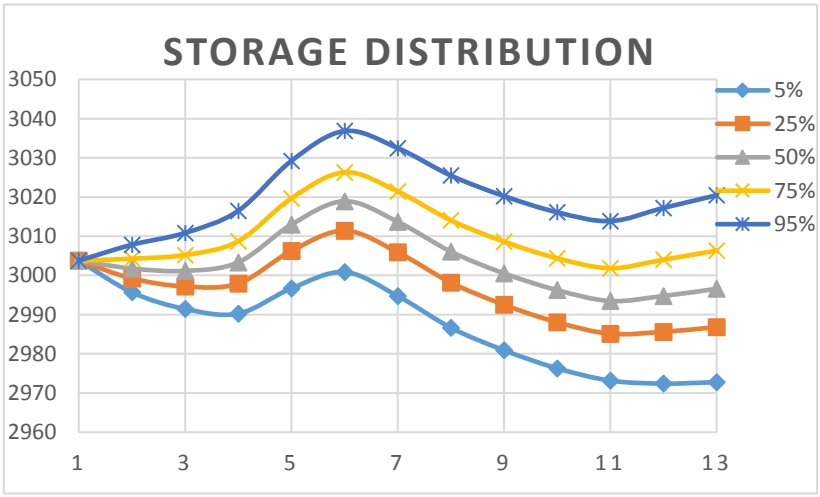
$$\mathrm{Q} = 135.147 \times [-397.658 - 0.00155\ \mathrm{S} + 50.63\ \ln(\mathrm{S})]^{1.679} \times \frac{bcm/mon}{cm/sec}$$

Substitute Q into S we can get S

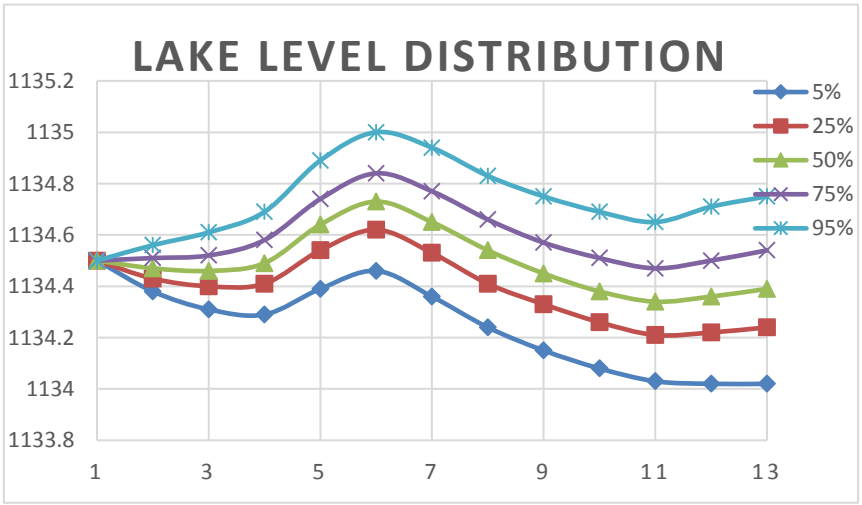
$$S_i = \left[S_{i-1} - 135.147 \times [-397.658 - 0.00155\ \mathrm{S} + 50.63\ \ln(S_{i-1})]^{1.679} \times \frac{bcm/mon}{cm/sec}\right] + W_{i-1} - D_{i-1}$$

Start from S_1 = 3003.76;

Get S distribution by code in **Appendix A**



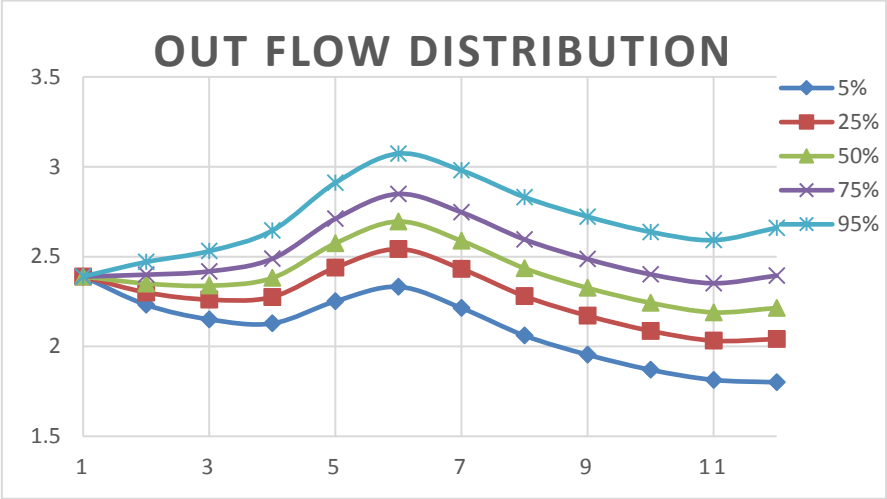
From $\mathrm{H} = 733.73 - 0.00155\ \mathrm{S} + 50.63\ \mathrm{lnS}$, we can derive H distribution, code is in **Appendix A.**



(a2) The probability H below 1133.5 at each month is 0 through 1 to 12.
Code is in **Appendix A**

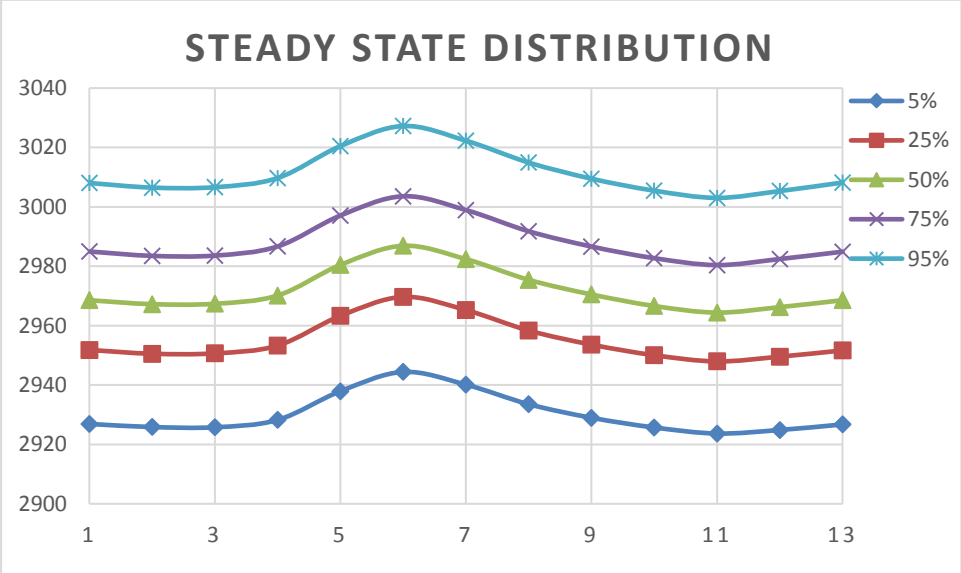
(a3) We can derive outflow by the function Q(S), code is in **Appendix A**

$$\mathrm{Q} = 135.147 \times [-397.658 - 0.00155\ \mathrm{S} + 50.63\ \ln(\mathrm{S})]^{1.679} \times \frac{bcm/mon}{cm/sec}$$



(a4) and (a5)

By calculating the 20 following years' distribution (code is in **Appendix A**), I find the distribution converge with the increase of time.



By comparing with the coming year(a1), we can conclude that the coming year is a wet year.

Appendix A:

//Calculating S distribution by formula.

```
//calculate 12 months distribution
for(i=0; i<12; i++)
{
    //calculate std deviation of D
    g[i] = sqrt(pow(0.1,2.0)*(1-pow(-0.75,2.0)));

    //Calculate 50000 trails of W
    w = randn(m[i], n[i]);

    for(int j=0; j<iter; j++)
    {
        f[j] = 0.5+((-0.75)*0.1/n[i])*(w[j]-m[i]);
        //calculate D's value
        //mean + stdvar*(12*ran - 6)
        d[j] = f[j] + g[i]*(ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()-6);

        q[i][j] = 135.147*pow((733.73-0.00155*s[i][j]+50.63*log(s[i][j])-1131.388) ,1.679)*(365*24*3600/(12*pow(10, 9)));
        s[i+1][j] = w[j] - d[j] + s[i][j] - q[i][j];
    }
    free(w);
}
```

//All results are sort increasingly and pick 5th, 25th, 50th, 75th, 95th.

```
void outResult(double **s){
    for(int i=0;i<13;i++)
    {
        sort(s[i], s[i]+iter);
        output<<s[i][iter/20]<<","<<s[i][iter/4]<<","<<s[i][iter/2]<<","<<s[i][iter*3/4]<<","<<s[i][iter*19/20]<<"\n";
    }
    output<<"\n";
}
```

//Calculate H distribution by function h(s) provided.

```
double **lakeLevel(double **s)
{
    double **h;
    h = (double**)malloc(13*sizeof(double*));

    for(int k=0;k<13;++k){
        h[k] = (double*)malloc(iter*sizeof(double));
    }
    for(int i=0;i<iter;i++){
        h[0][i] = 1134.5;
    }
    for(int i=1;i<13;i++){
        for(int j=0;j<iter;j++){
            h[i][j] = 733.73-0.00155*s[i][j]+50.63*log(s[i][j]);
        }
    }
    return h;
}
```

//Calculate probability below H=1133.5

```
void outBelow(double **h)
{
    for(int i=0;i<13;i++)
    {
        int c=0;
        for(int j=0;j<iter;j++){
            if(h[i][j]<1133.5){
                c++;
            }
        }
        output<<c/iter<<"\n";
    }
    output<<"\n";
}
```

//Calculate Q distribution

```
double **outflow(double **h)
{
    double **q;
    q = (double**)malloc(12*sizeof(double*));

    for(int k=0;k<12;++k){
        q[k] = (double*)malloc(iter*sizeof(double));
    }

    for(int i=0;i<12;i++){
        for(int j=0;j<iter;j++){
            q[i][j] = 135.147*pow((h[i][j]-1131.388) ,1.679)*(365*24*3600/(12*pow(10, 9)));
        }
    }
    return q;
}
```

//Calculate 1st year's future storage (base condition for recursion)

```
double **futureStorage(double **p, int year)
{
    double stdvar=0;
    if(year==0)
    {
        //Calculate the distribution of year 1
        output<<year+1<<" year Storage Distribution\n";
        outResult(p);
        for(int i=0;i<12;i++){
            double sum=0;
            for(int j=0;j<iter;j++){
                sum = sum + pow((p[i+1][j]-p[i+1][iter/2]),2);
            }
            stdvar = sqrt(sum/iter);
            output<<"Mean,"<<p[i+1][iter/2]<<",stdvar,"<<stdvar<<"\n";
        }
        output<<"\n";
        return p;
    }
```

//Calculate following years by recursion.

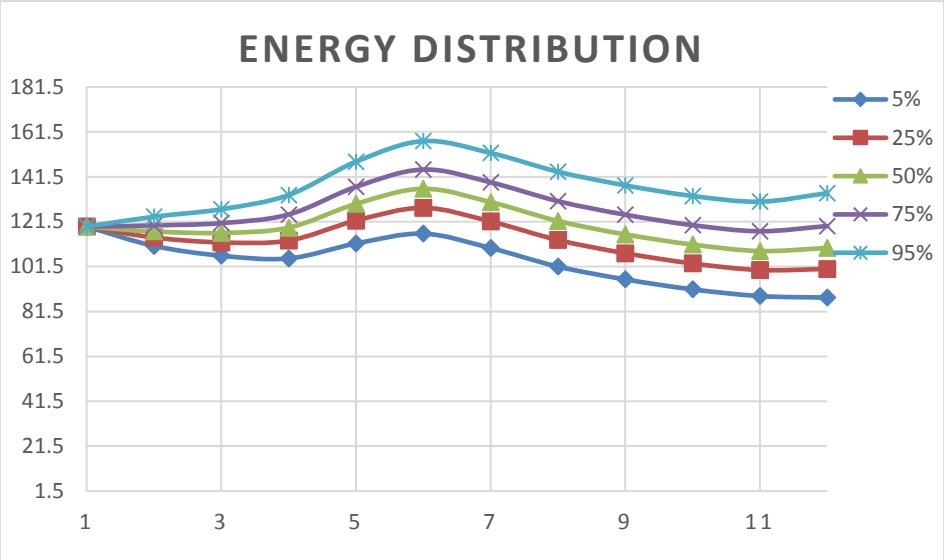
```
else
{
    double** pre;
    pre = futureStorage(p, year-1);
    //Input 12 months mean and std deviation for W
    double m[12] = {0.9, 2.3, 5.0, 12.5, 9.0, -2.0, -4.5, -2.6, -1.5, 0.0, 4.0, 4.5};
    double n[12] = {3.6, 4.6, 5.4, 6.0, 5.0, 4.0, 3.5, 3.0, 3.0, 3.5, 6.0, 5.5};
    //Define mean and std deviation for D
    double f[iter], g[12];

    //Calculating Storage of 12 month
    double *w;
    double **s, **q;
    s = (double**)malloc(13*sizeof(double*));
    q = (double**)malloc(13*sizeof(double*));

    for(int k=0;k<13;++k){
        s[k] = (double*)malloc(iter*sizeof(double));
        q[k] = (double*)malloc(iter*sizeof(double));
    }
    //initiate first month beginning storage
    for(int i=0;i<iter;i++)
        s[0][i] = pre[12][i];

    double d[iter];
```

```
//calculate 12 months distribution
for(int i=0; i<12; i++)
{
    //calculate std deviation of D
    g[i] = sqrt(pow(0.1,2.0)*(1-pow(-0.75,2.0)));

    //Calculate 50000 trails of W
    w = randn(m[i], n[i]);

    for(int j=0; j<iter; j++)
    {
        f[j] = 0.5+((-0.75)*0.1/n[i])*(w[j]-m[i]);

        d[j] = f[j] + g[i]*(ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()-6);

        q[i][j] = 135.147*pow((733.73-0.00155*s[i][j]+50.63*log(s[i][j])-1131.388) ,1.679)*(365*24*3600/(12*pow(10, 9)))
        s[i+1][j] = w[j] - d[j] + s[i][j] - q[i][j];
    }
    free(w);
}
```

(b1)

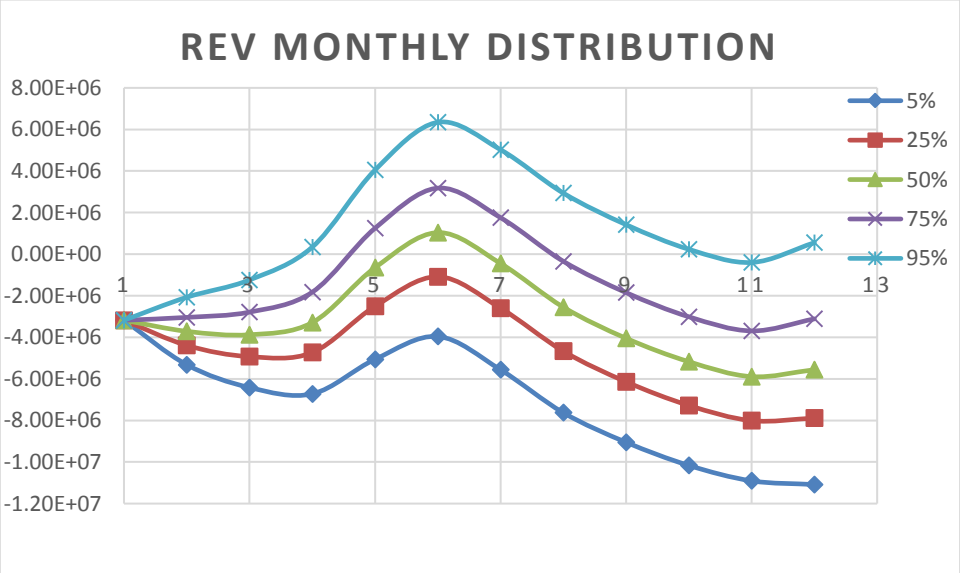
By function, $\mathrm{Ei} = 0.00706\,\eta\,[\mathrm{H} - 1115] \times Q_i(s)$, calculate E. Code is in **Appendix B**

We get distribution of E



(b2)

i. Monthly net revenue distribution for each of 12 months is calculated Code is in **Appendix B**



ii. Expected annual net revenue = -3.72E+07

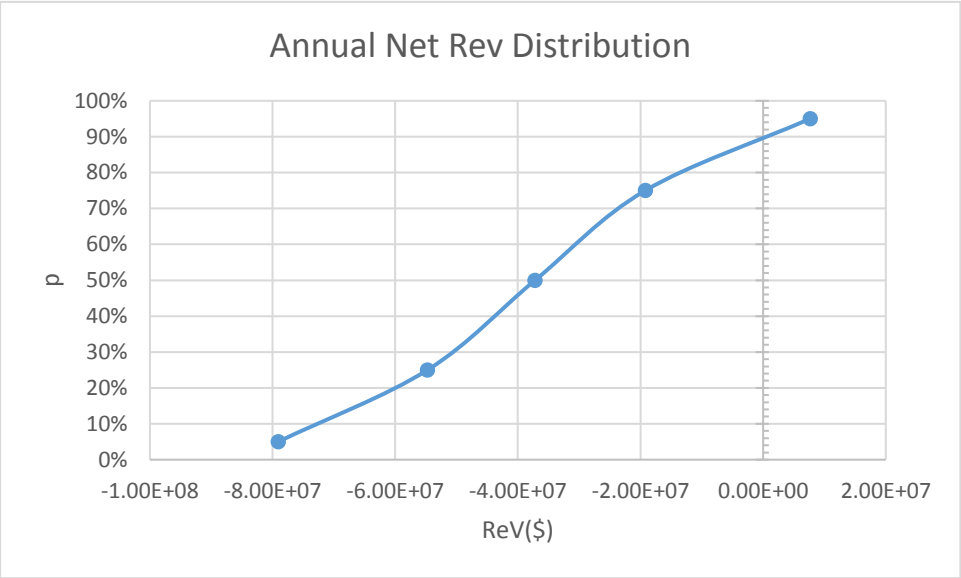
iii. Tariff that make mean 0:

$\mathrm{x} \times 12 \times 220 + \text{net revenue} = 0$

x= 14087.3

So the tariff that make mean 0 is 114087.3 $/GWh

iv. Annual Net Revenue: (Code is in **Appendix B**)



v. Make 25% zero:

$$x \times 12 \times 220 + \text{net revenue} = 0$$

x= 20731.1

So the tariff that make 25% 0 is 120731.1 $/GWh

Comparing 50% value, to ensure 25% net revenue to be 0, we need charge more.

Because now cdf increases with annual rev.

Appendix B //Calculate monthly energy distribution for 100,000 times

```
double **energy(double **h, double **q)
{
    double **e;
    e = (double**)malloc(12*sizeof(double*));

    for(int k=0;k<12;++k){
        e[k] = (double*)malloc(iter*sizeof(double));
    }
    for(int i=0;i<12;i++){
        for(int j=0;j<iter;j++){
            e[i][j] = 0.00706*0.94*(h[i][j]-1115)*q[i][j]*(pow(10,9)/(24*(365/12)*3600));
        }
    }
    return e;
}
```

//calculate net revenue monthly

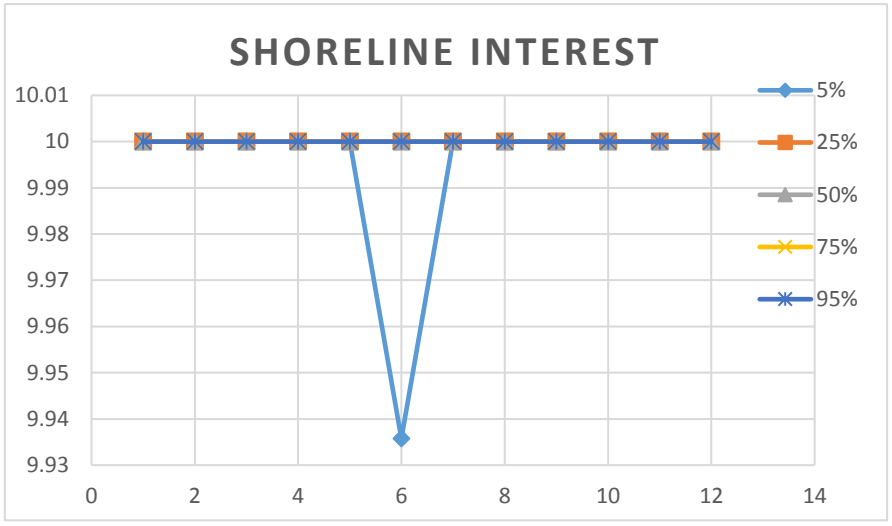
```
double **netRev(double **e)
{
    double **r;
    r = (double**)malloc(12*sizeof(double*));

    for(int k=0;k<12;++k){
        r[k] = (double*)malloc(iter*sizeof(double));
    }
    for(int i=0;i<12;i++){
        for(int j=0;j<iter;j++){
            if(e[i][j]<220){
                r[i][j]=100000*220-250000*(220-e[i][j]);
            }
            else{
                r[i][j]=100000*220;
            }
        }
    }
    return r;
}
```

//calculate annual revenue

```
double *anualRev(double **r)
{
    double *a;
    a = (double*)malloc(iter*sizeof(double));
    for(int i=0;i<iter;i++){
        a[i]=0;
        for(int j=0;j<12;j++){
            a[i] = a[i] + r[j][i];
        }
    }
    return a;
}
```

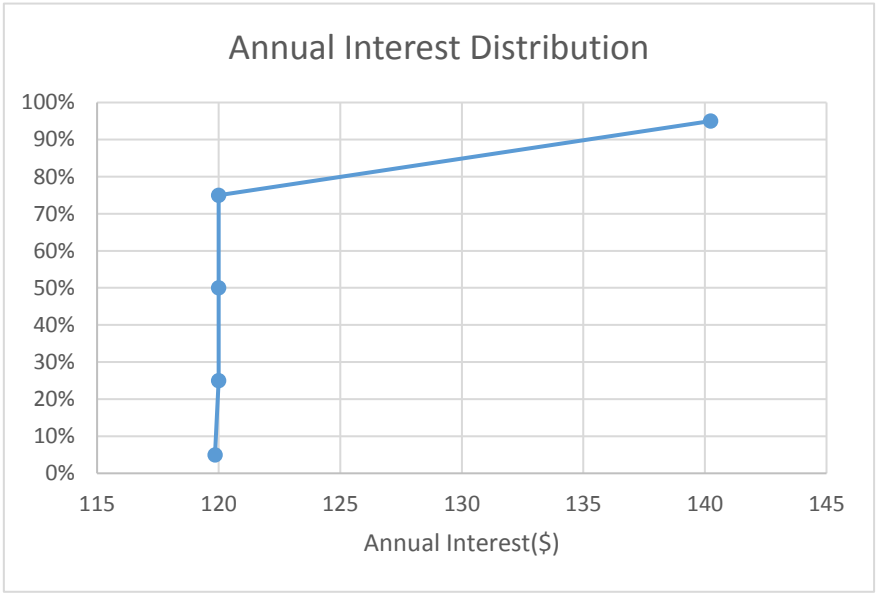
```
void outAnual(double *a)
{
    std::sort(a,a+iter);
    output<<a[iter/20]<<","<<a[iter/4]<<","<<a[iter/2]<<","<<a[iter*3/4]<<","<<a[iter*19/20]<<"\n\n";
}
```

(c1) Shoreline economic value can be calculated by lake level. Code is in **Appendix C**.



The 5th to 95th value all equals to 10 except the 5th for month 6.

(c2)/(c3)



From cdf plot we can see mean annual interest is 120 $.

Appendix C //Calculate shoreline economic Interest:

```
double **hInter(double **h)
{
    //Shoreline interest related to H
    double **t;
    t = (double**)malloc(12*sizeof(double*));

    for(int k=0;k<12;++k){
        t[k] = (double*)malloc(iter*sizeof(double));
    }
    for(int i=0;i<12;i++){
        for(int j=0;j<iter;j++){
            if(h[i][j]<1134){
               t[i][j] = 10*h[i][j]-11330;
            }
            if(h[i][j]>=1134 && h[i][j]<=1135){
                t[i][j] = 10;
            }
            else{
                t[i][j] = -20*h[i][j]+22710;
            }
        }
    }
    return t;
}
```

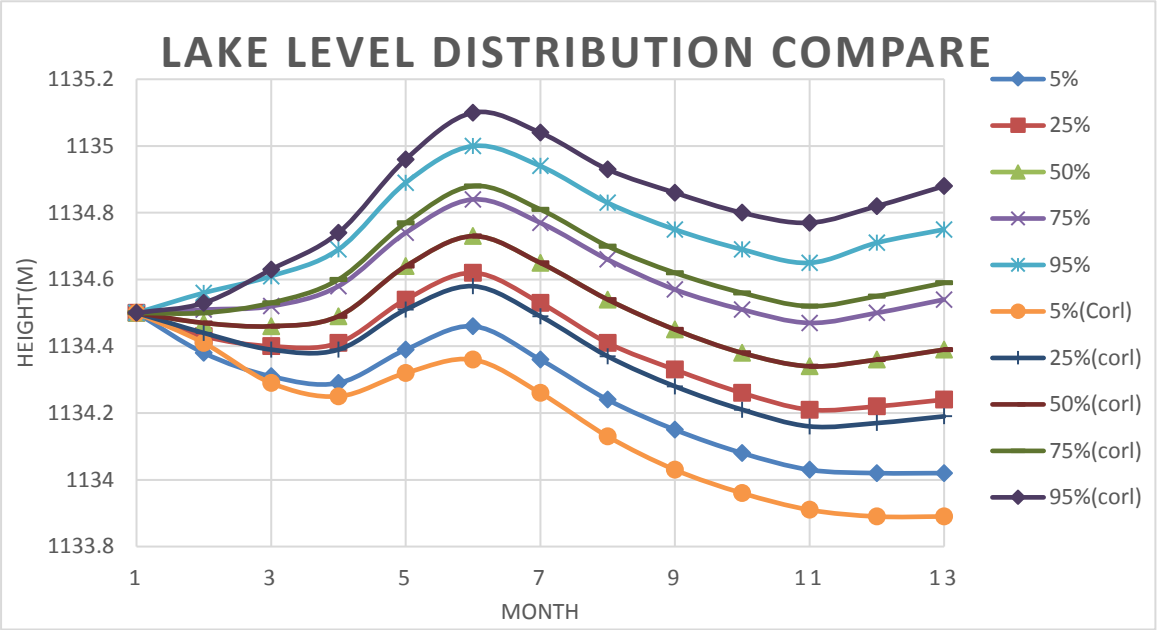
(d) Resolve problems. (Codes are presented in (**Appendix D**)

Apply a correlation between month-to-month inflow.

The key point is to calculate the mean and std variation of D, W

We use bivariate normal random numbers generating function:

```
  //calculate std deviation of D
  g[i] = sqrt(pow(0.1,2.0)*(1-pow(-0.75,2.0)));
  n[i] = sqrt(pow(n[i],2.0)*(1-pow(0.75,2.0)));

//Calculate iter trails of W
w[j] = m[i][j] + n[i]*(ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()-6);
if(i<11){
    m[i+1][j] = m[i+1][j]+0.75*(n[i+1]/n[i])*(w[j]-m[i][j]);
}
//Calculate mean of j th D
f[j] = 0.5+((-0.75)*0.1/n[i])*(w[j]-m[i][j]);
```

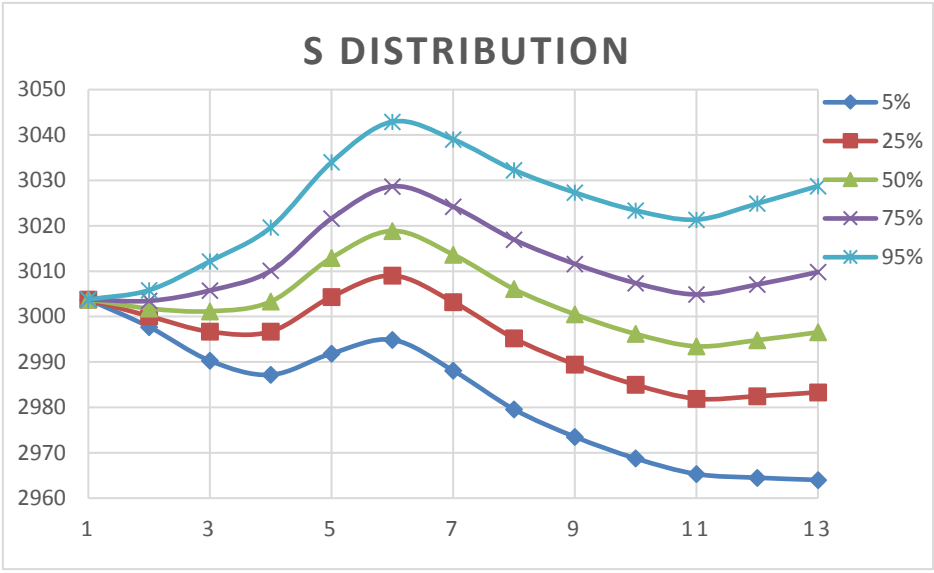


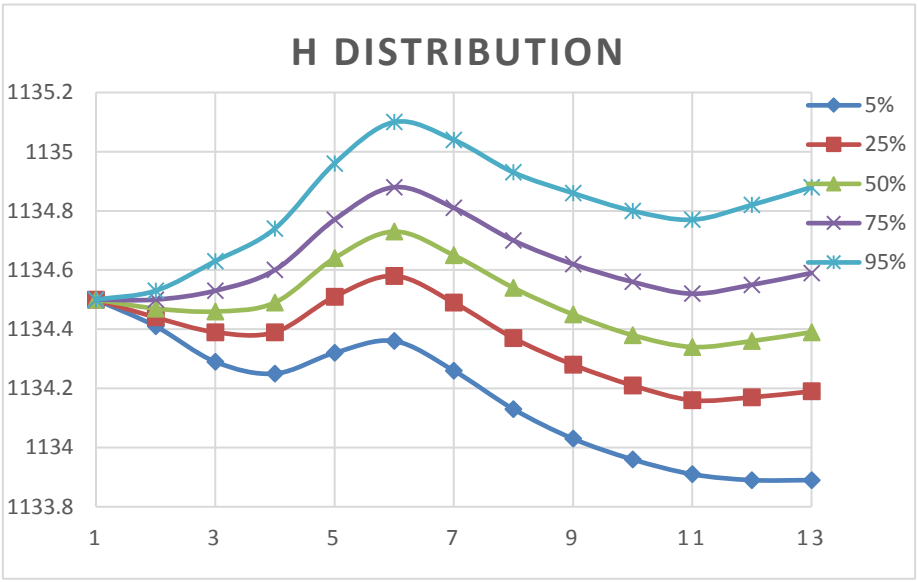
As shown in the chart, when the distribution below mean, the estimated value for every month with correlation is lower;

When the distribution at mean, the estimated value for every month with correlation equals the value without correlation;

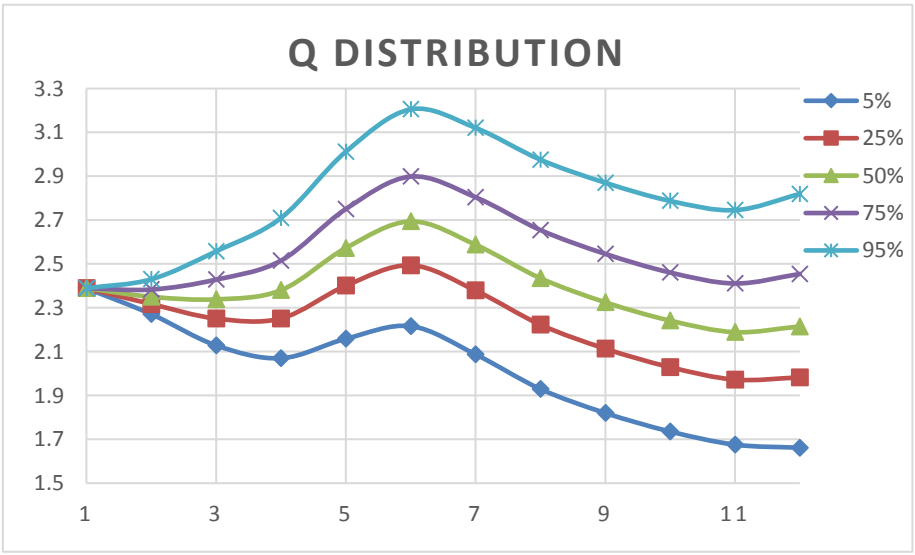
When the distribution above mean, the estimated value for every month with correlation is higher.

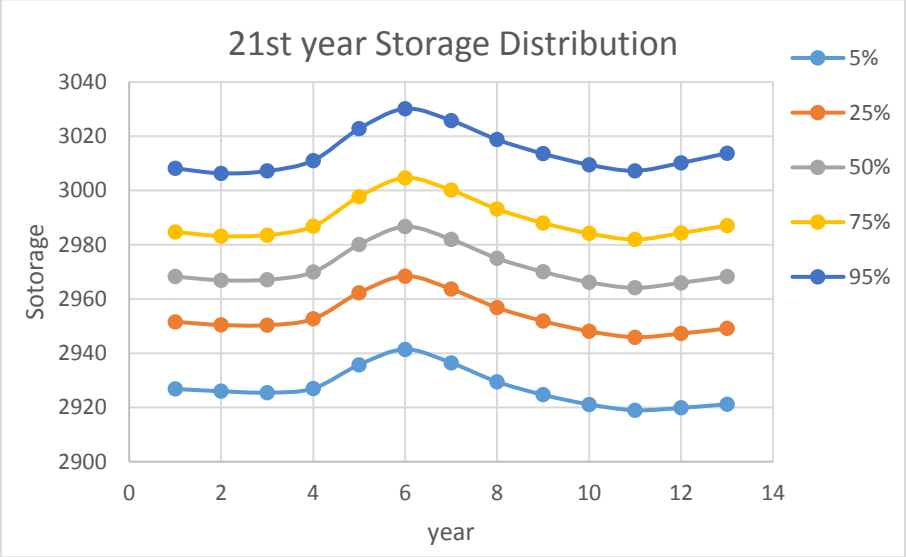
(a)





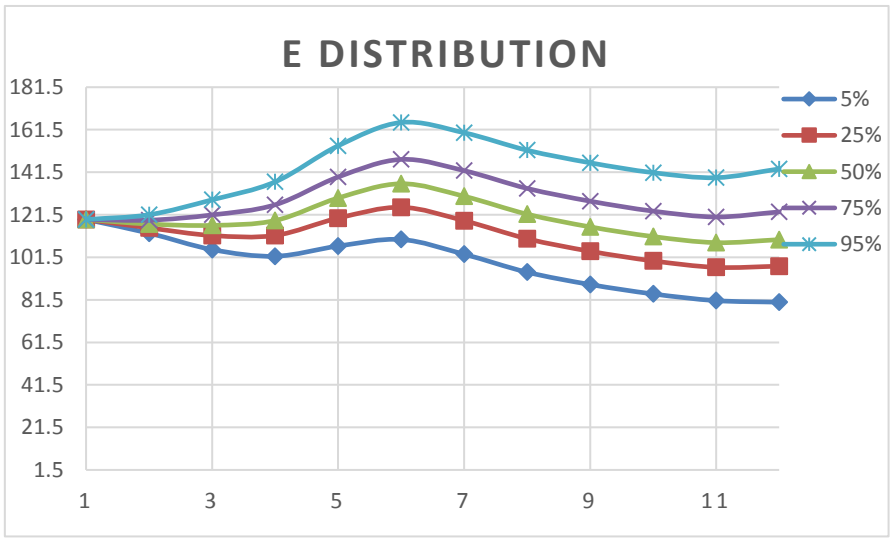
The probability for each month less than 1133.5 is 0.



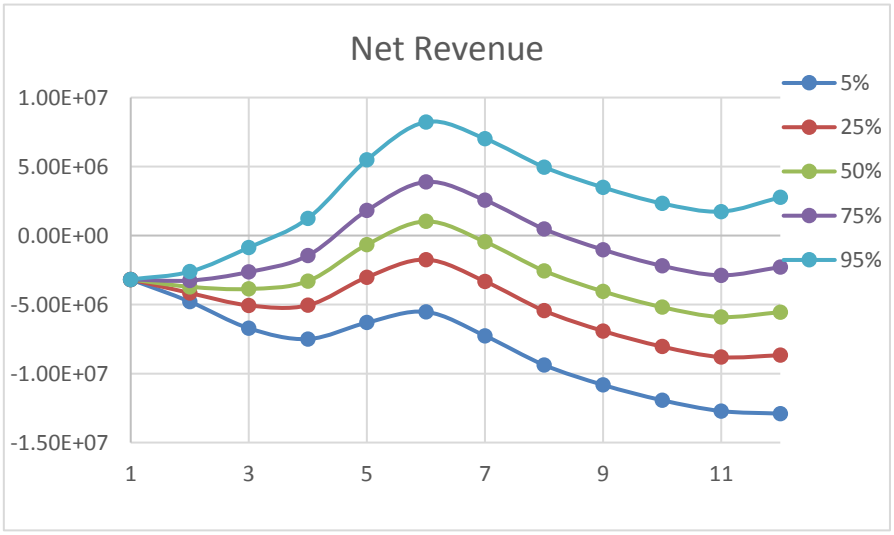


The chart above shows steady state distribution on 21st year.

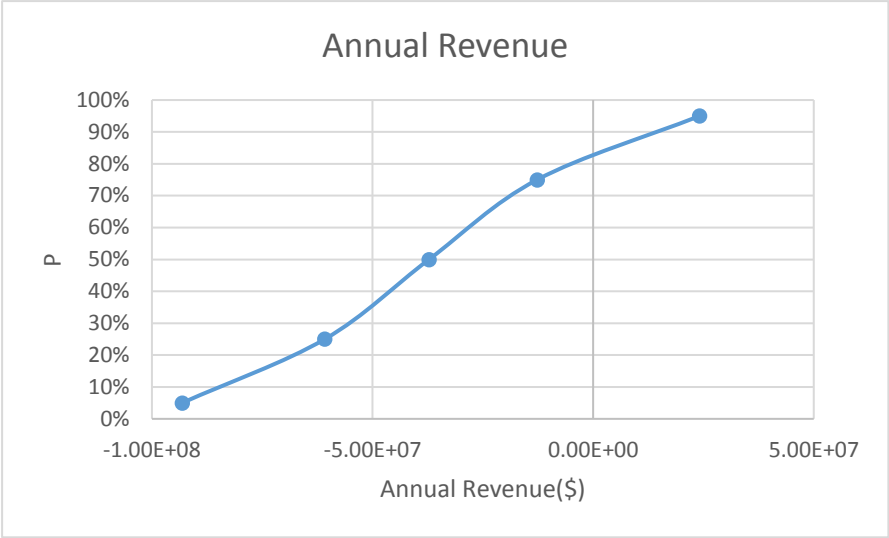
(b)



i.



ii.



iii. Expected annual net revenue = -3.72E+07

iv. Tariff that make mean 0:

$$x \times 12 \times 220 + \text{net revenue} = 0$$

x= 14089.3

So the tariff that make mean 0 is 114089.3 $/GWh

v. Tariff that makes 25% 0:

$$x \times 12 \times 220 + \text{net revenue} = 0$$

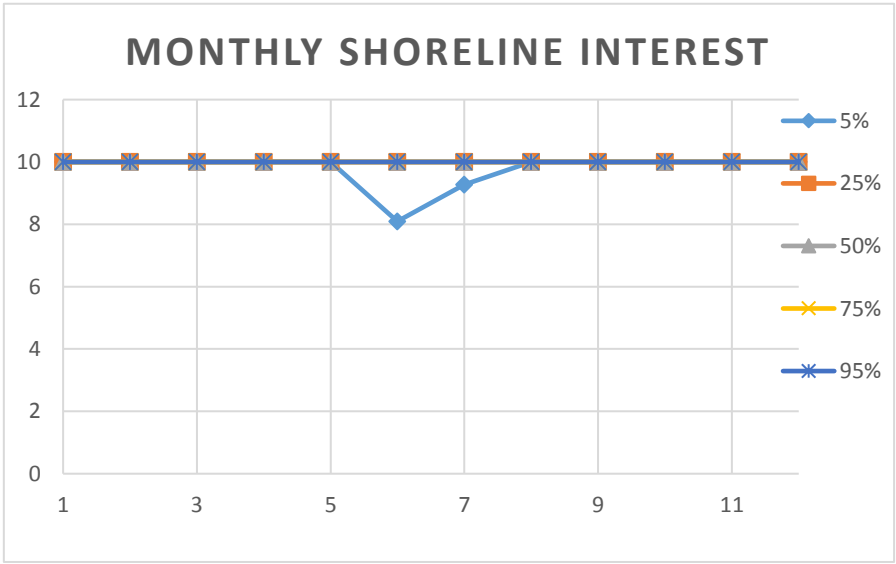
x= 23041.1

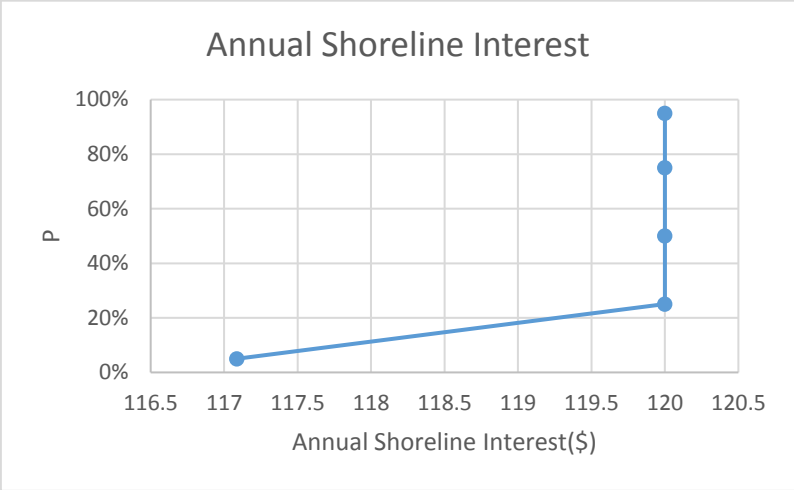
So the tariff that make 25% 0 is 123041.1 $/GWh

Comparing 50% value, to ensure 25% net revenue to be 0, we need charge more.

Because now cdf increases with annual rev.

(c)

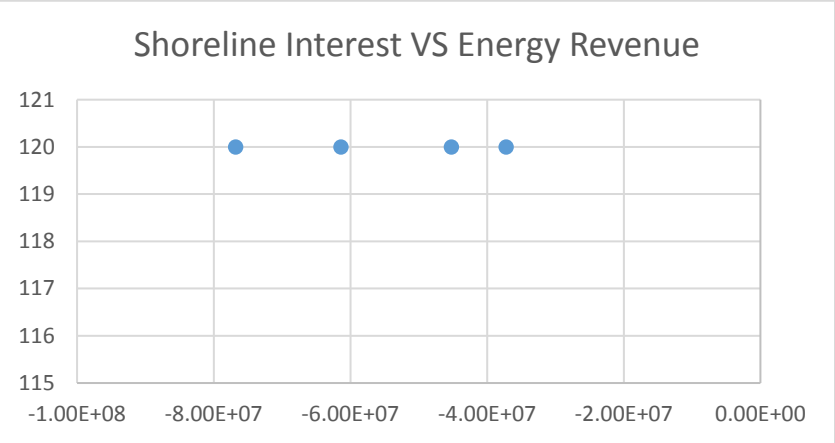




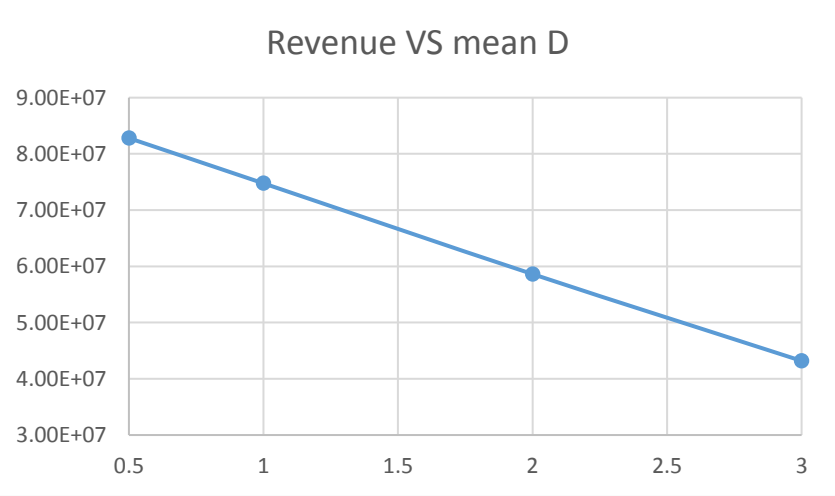
From cdf plot we can see mean annual interest is 120 $.

(e1)

i. Shoreline Interest VS Energy Revenue



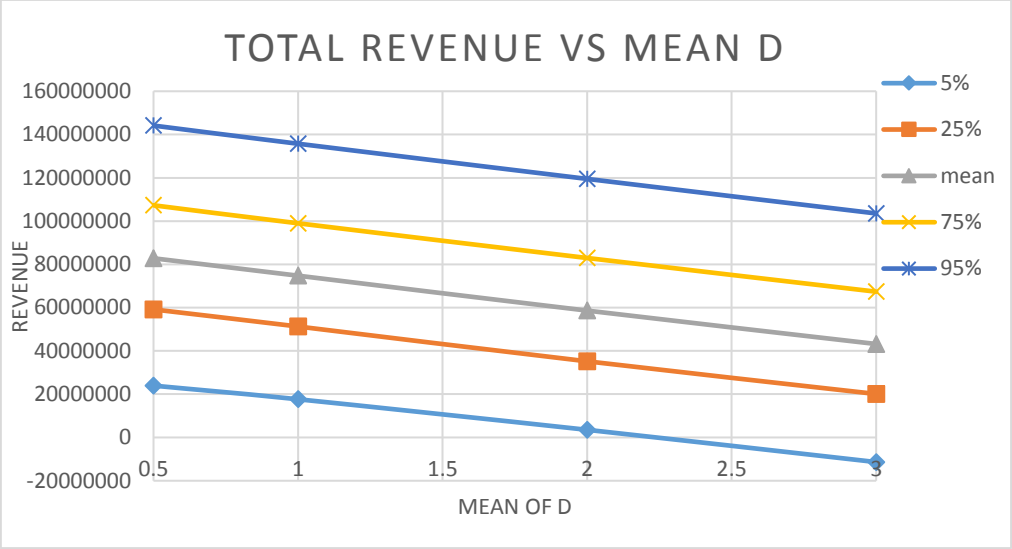
ii. Expected annual energy revenue VS mean D:



iii. Uganda should convince the upstream countries to establish a limit on withdrawal in the basin.

The limit should be 0.5 bcm/month, as low as possible.

(e2) Expected annual energy revenue VS D:



The limit should be as low as possible.

Appendix D //Input for storage calculation

```
double **reStorage()
{
    double **m;
    m = (double**)malloc(12*sizeof(double*));

    for(int k=0;k<12;++k){
        m[k] = (double*)malloc(iter*sizeof(double));
        m[k] = (double*)malloc(iter*sizeof(double));
    }
    //Input 12 months mean and std deviation for W
    for(int j=0;j<iter;j++){
        m[0][j]=0.9; m[1][j]=2.3; m[2][j]=5.0; m[3][j]=12.5; m[4][j]=9.0; m[5][j]=-2.0;
        m[6][j]=-4.5; m[7][j]=-2.6; m[8][j]=-1.5; m[9][j]=0; m[10][j]=4; m[11][j]=4.5;
    }
    double n[12] = {3.6, 4.6, 5.4, 6.0, 5.0, 4.0, 3.5, 3.0, 3.0, 3.5, 6.0, 5.5};
    //Define mean and std deviation for D
    double f[iter], g[12];

    //Declare Storage and outflow of 12 month
    double **s, **q;
    s = (double**)malloc(13*sizeof(double*));
    q = (double**)malloc(13*sizeof(double*));
    for(int k=0;k<13;++k){
        s[k] = (double*)malloc(iter*sizeof(double));
        q[k] = (double*)malloc(iter*sizeof(double));
    }
```

//initiate first month beginning storage

```
//initiate first month beginning storage
int i;
for(i=0;i<iter;i++)
    s[0][i] = 3003.76;

//declare mean of D
double d[iter];

//calculate 12 months distribution
for(i=0; i<12; i++)
{
    //printf("hello ");
    //calculate std deviation of D
    g[i] = sqrt(pow(0.1,2.0)*(1-pow(-0.75,2.0)));
    n[i] = sqrt(pow(n[i],2.0)*(1-pow(0.75,2.0)));

    //Declare net inflow W for a month
    double *w;
    w = (double*)malloc(iter*sizeof(double));
```

//Calculate storage distribution

```
for(int j=0; j<iter; j++)
{
    //Calculate iter trails of W
    w[j] = m[i][j] + n[i]*(ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()-6);
    if(i<11){
        m[i+1][j] = m[i+1][j]+0.75*(n[i+1]/n[i])*(w[j]-m[i][j]);
    }
    //Calculate mean of j th D
    f[j] = 0.5+((-0.75)*0.1/n[i])*(w[j]-m[i][j]);
    //calculate D's value
    //mean + stdvar*(12*ran - 6)
    d[j] = f[j] + g[i]*(ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()+ran()-6);

    q[i][j] = 135.147*pow((733.73-0.00155*s[i][j]+50.63*log(s[i][j])-1131.388) ,1.679)*(365*24*3600/(12*pow(10, 9)));
    s[i+1][j] = w[j] - d[j] + s[i][j] - q[i][j];
}
```

//Related calculations for H, D, Rev use similar code to part a, b, c.