A Project Report on

Predicting Software reliability using Machine learning

Project report submitted in partial fulfilment for the requirement of award of B.Tech. Degree in Information Technology

By

Makesh Kumar M 116015083

Avinash R 116015024

Under the Guidance of
Prof. N.Madhu Sudanan Rao
Assistant Professor-II
School of Computing

SCHOOL OF COMPUTING
SHANMUGHA
ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY
(SASTRA UNIVERSITY)
(A University Established under section 3 of the UGC Act, 1956)
TIRUMALAISAMUDRAM
THANJAVUR – 613401
April 2016

SHANMUGHA

ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY
(SASTRA UNIVERSITY)
(A University Established under section 3 of the UGC Act, 1956)
TIRUMALAISAMUDRAM
THANJAVUR – 613401

SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that the Project entitled
Predicting Software reliability using Machine learning

is a work done by

Makesh Kumar M 116015083
Avinash R 116015024

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY
OF SASTRA UNIVERSITY, Thanjavur during the year 2014-15

Internal Guide **Associate Dean**
Department of Information Technology

Submitted for the university examination held on:

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT:

First and Foremost, We take pride in thanking the Almighty who gave us strength for the successful completion of this project.

We take immense pride in thanking our parents and friends who were a constant source of encouragement and inspiration and provided their cooperation to enable us to complete this project to the fullest satisfaction..

We are extremely thankful to my dignified Dean, **Dr.P.Swaminathan**, School of Computing and Associate Dean,**Dr.N.Sairam** for inspiring and motivating me to bring out a perfect and successful project work.

I am also very grateful to **Prof N. Madhu Sudanan Rao**, Assistant Professor- II for being very helpful with coordinating among the students and providing useful tips.

Last but not least, I also thank all the Teaching, Non-Teaching staff, and those who have directly or indirectly helped me by extending their moral support and encouragement for completion of this project.

TABLE OF CONTENTS

SYNOPSIS

The current society relies on the software systems. Hence, the reliability of software must be improved. Software Reliability is the probability that a system will function properly under a given environment. It is calculated during the testing phase of the software itself by using the large amount of testing datasets. Maximum likelihood estimation (MLE) equation is solved to obtain estimates, which are used for calculating reliability. The BAT algorithm is used to predict the reliability of software. The BAT algorithm is based on the echolocation capability of bats. The natural properties of bats such as velocity, population and loudness are considered as attributes for the algorithm. The value obtained through BAT can be used only for low dimensional problems. The BAT algorithm has poor local and global search characteristics for complex problems.

Thus an Enhanced BAT (EBAT) algorithm is used.BAT is bad at Exploration and Exploitation. This problem has been resolved by introducing two modifications to the original BAT algorithm. The two modifications are i) Inertia weight modification
ii) Distribution of population modification.

In this project, we compare the reliability calculated by employing traditional method, BAT algorithm and Enhanced BAT algorithm. The EBAT guaranteed better reliability compared to traditional method and BAT algorithm. The results show that the EBAT algorithm is superior to traditional and BAT algorithm.

System Configuration

Hardware Requirements:

The hardware requirements of the current project are

- RAM: Minimum of 1GB of RAM required
- Disk space:1GB of MATLAB setup and Minimum of 4GB for installation
- Processor: Any Intel or AMD x86 processor supporting SSE2 instruction set

Software Requirements:

The software requirements for the current project

Operating Systems

- Windows: Windows XP with service pack 3, Windows Vista with service pack 2, windows 7 with service pack 1, windows 8 and above
- Linux: Ubuntu 14.04 LTS and 12.10, Red Hat Enterprise Linux 5.x and 6.x and Qualified distributions

MATLAB

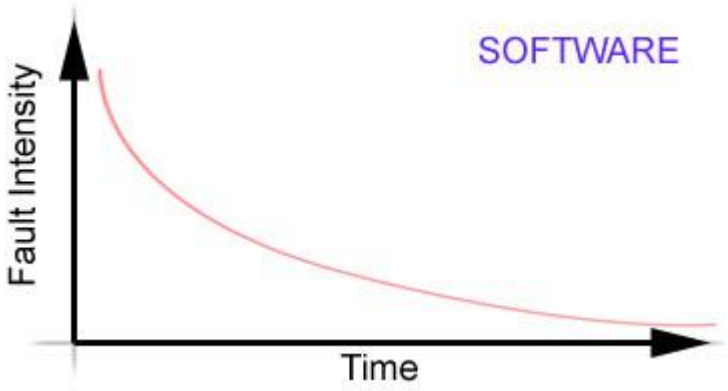
- MATLAB 8.1.0.604 (R2013a) for 64 bit is used
- Required 32 or 64 bit version of MATLAB 2012a and above for windows 64 bit version MATLAB 2012a version and above for Linux

PROJECT DESCRIPTION

The application of computer software has crossed into many different fields, with software being an essential part of industrial, commercial and military systems. Because of its many applications in safety critical systems, software reliability is now important research area. Software Reliability is the probability that software will work properly in a specified environment and for a given amount of time. Although software engineering is becoming the fastest developing technology of the last century, there is no complete, scientific, quantitative measure to assess them. Software reliability testing is being used as a tool to help assess these software engineering technologies.

To improve the performance of software product and software development process, a thorough assessment of reliability is required. Software reliability is important because it is of great use for software managers and practitioners.

Software reliability is measured in terms of mean time between failures (MTBF). Mean time between failures (MTBF) describes the expected time between two failures for a repairable system. Reliability for good software is a number between 0 and 1. Reliability increases when errors or bugs from the program are removed.



Hardware contains decreasing and increasing failure rates. The primary reason for increasing failure rate is due to hardware component wear out or aging. But, the software exhibits a decreasing failure rate. Thus analytical models have been used to address the problem of software reliability measurement. The models are based mainly on the failure history of the software.

In this project, we use a fault count model. This class of model is concerned with modelling the number of failures seen or faults detected in given testing intervals. Musa Execution Time model is a type in this class has been used to estimate the reliability. This model has the widest distribution among the software reliability models and was developed by John Musa**.**

Assumptions of Musa model:

- The rate of fault detection is directly proportional to the current fault content of the software
- The fault detection rate remains constant over the intervals between fault occurrence
- A fault is corrected instantaneously without introducing new faults into the software
- The software is operated in a similar manner as that in which reliability predictions are to be made
- The failure when the faults are detected are independent

In this project, we use a concept called Maximum likelihood estimation to find the estimates. In statistics, maximum-likelihood estimation (MLE) is a method of estimating the parameters of a statistical model given data. In general, for a fixed set of data and underlying statistical model, the method of maximum likelihood selects the set of values of the model parameters that maximizes the Likelihood Function.

Model estimation:

Suppose we have observed 'n' failures of the software system at time $t_1, t_2, t_3, \ldots, t_n$ and from the last failure time t_n an additional time of $x(x \geq 0)$ has elapsed without failure. Then t_n+x is the total time the software component has been observed since the start. Using the model assumptions the Likelihood function for Musa model is obtained as

$$L(\beta_0, \beta_1) = \beta_0^n \beta_1^n \left[\prod_{i=1}^{n} \exp(-\beta_1 t_i) \right] \exp(-\beta_0[1 - \exp(-\beta_1(t_n + x))])$$

So the MLEs of β_0 and β_1 are obtained as the solutions of the following pairs of equations:

$$\hat{\beta}_0 = \frac{n}{1 - \exp(-\hat{\beta}_1(t_n + x))}$$

and

$$\frac{n}{\hat{\beta}_1} - \frac{n(t_n + x)}{\exp(\hat{\beta}_1(t_n + x)) - 1} - \sum_{i=1}^{n} t_i = 0$$

Once the estimates of $\boldsymbol{\beta_0}$ and $\boldsymbol{\beta_1}$ are obtained, we can estimate the other reliability measures. These include the reliability function and the failure intensity function.

The Failure Intensity function is given by

$$\lambda(t;\, \beta_0, \beta_1) = \beta_0\beta_1\exp(-\beta_1 t)$$

Reliability Calculation:

Suppose we denote $\mathbf{R(t+\Delta t|t)}$ as the conditional reliability function that the software will still operate after $\mathbf{t+\Delta t}$ given that it has not failed after time **t**. Let $\boldsymbol{\mu(t)}$ be the mean value function for the cumulative number of failures and $\boldsymbol{\lambda(t)}$ be the failure intensity function then

$$R(t + \Delta t \mid t) = \exp(-(\mu(t + \Delta t) - \mu(t)))$$

$$= \exp\left(-\int_t^{t+\Delta t} \lambda(x)\, dx\right)$$

using the mean value theorem of integrals

Dataset used in this project:

The software failure datasets are obtained from Musa. The following dataset has 100 observations of the pair (t, Y_t) pertaining to software failure.

Y_t represents the time to failure of the software after the t^{th} modification has been made.

Modification	Time between failures
0	5.7683
1	15.3426
2	24.4476
3	32.4131
4	41.0613
5	51.05

BAT Algorithm

BAT is a heuristic nature inspired algorithm. The algorithm works based on the echolocation capability of bats guiding them on the foraging behaviour. The echolocation property helps to find the distance between preys, similarly the BAT algorithm tends to find the high or optimum value in the search space. The global best solution is obtained after comparison of all solutions among N bats. The local search is provided in BAT algorithm by considering the loudness (A) of the bat. The loudness and pulse emission rate (r) are updated as the bat gets closer to the target.

Equations:

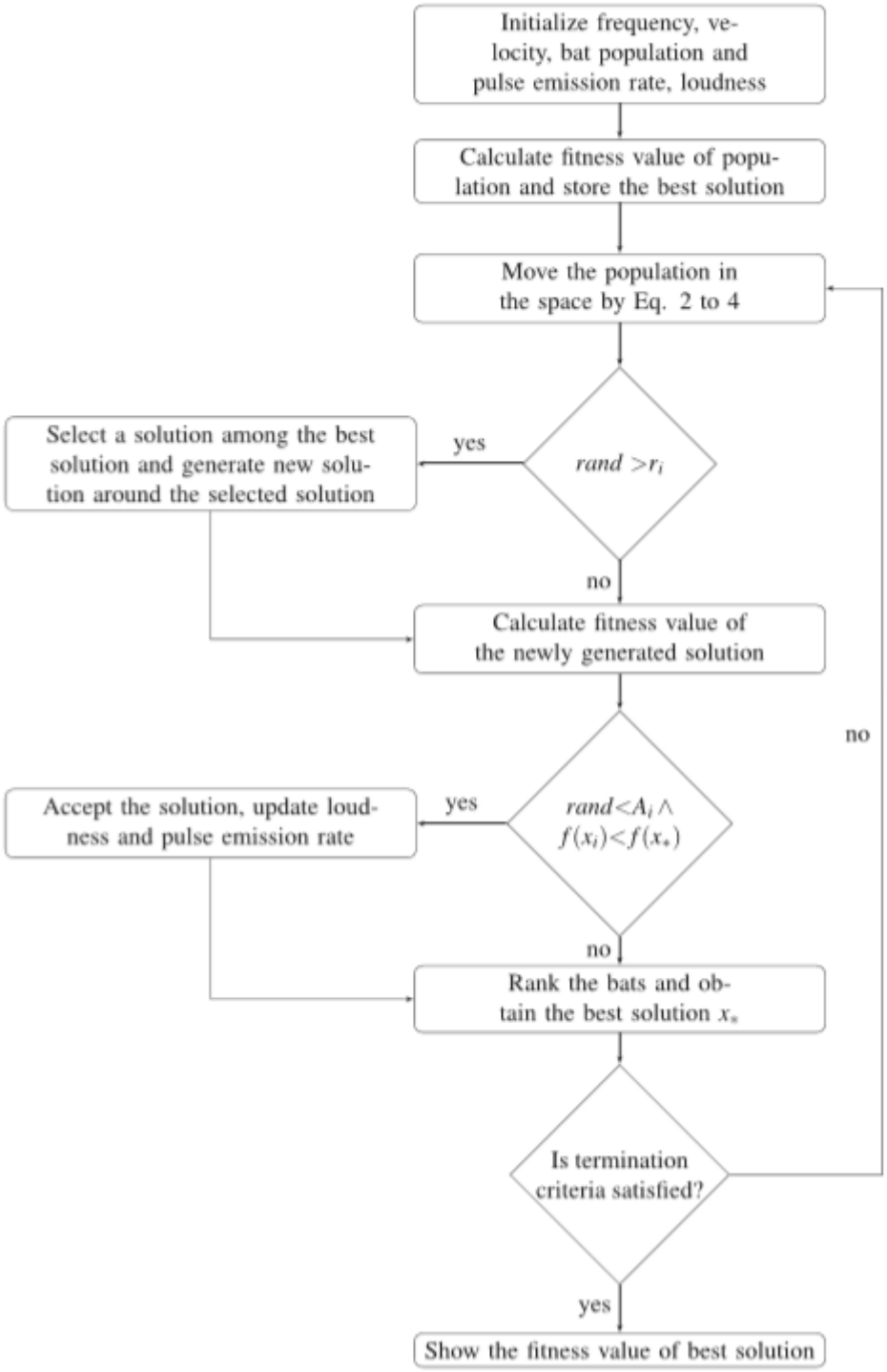
$$x_{ij} = x_{\min} + \varphi(x_{\max} - x_{\min}) \quad (1)$$

$$f_i = f_{\min} + \beta(f_{\max} - f_{\min}) \quad (2)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i \quad (3)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (4)$$

BAT algorithm flowchart:



EBAT algorithm

The BAT algorithm is bad at exploration and exploitation property. In order to tackle this problem two modifications have been added to the original BAT algorithm. The two modifications are: i) Inertia weight modification ii) Distribution of population modification

i) Inertia weight modification

$$v_i^t = \omega(v_i^{t-1}) + (x_i^t - x_*)f_i$$

Where ώ is the inertia weight factor that balances the global and local search intensity

ii) Distribution of population modification

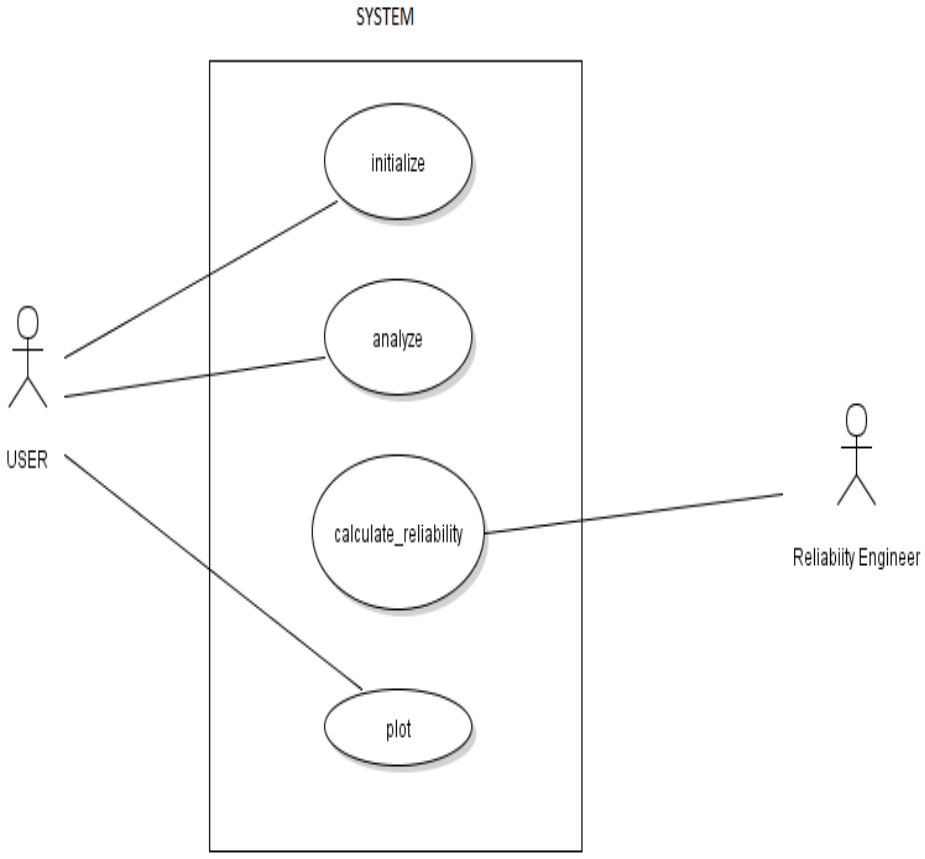
$$v_i^t = \omega(v_i^{t-1}) + (x_i^t - x_*)f_i\zeta_1 + (x_i^t - x_k^t)f_i\zeta_2$$

$$\zeta_1 + \zeta_2 = 1$$

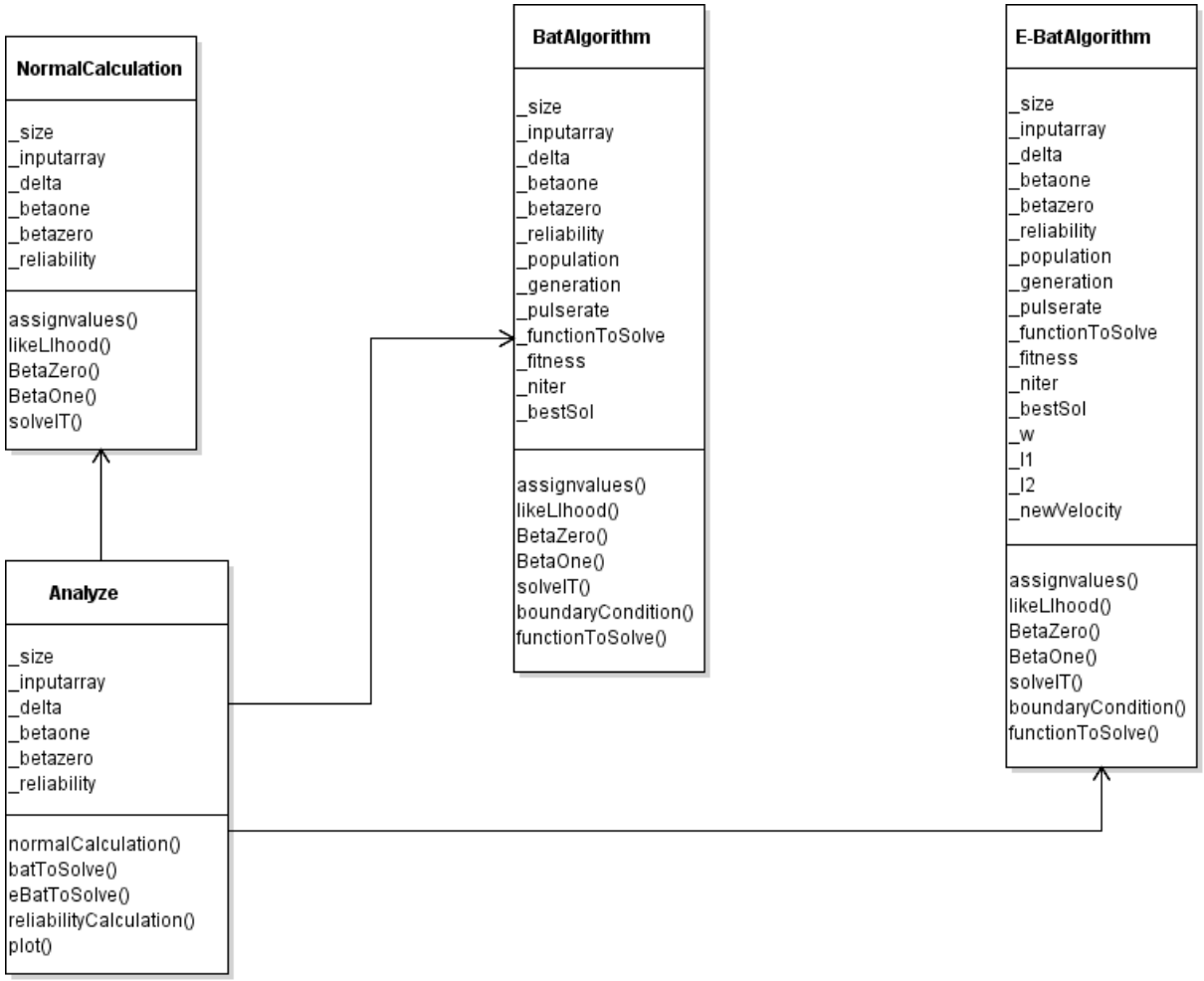
where ζ_1 is the learning factor

UML DIAGRAMS

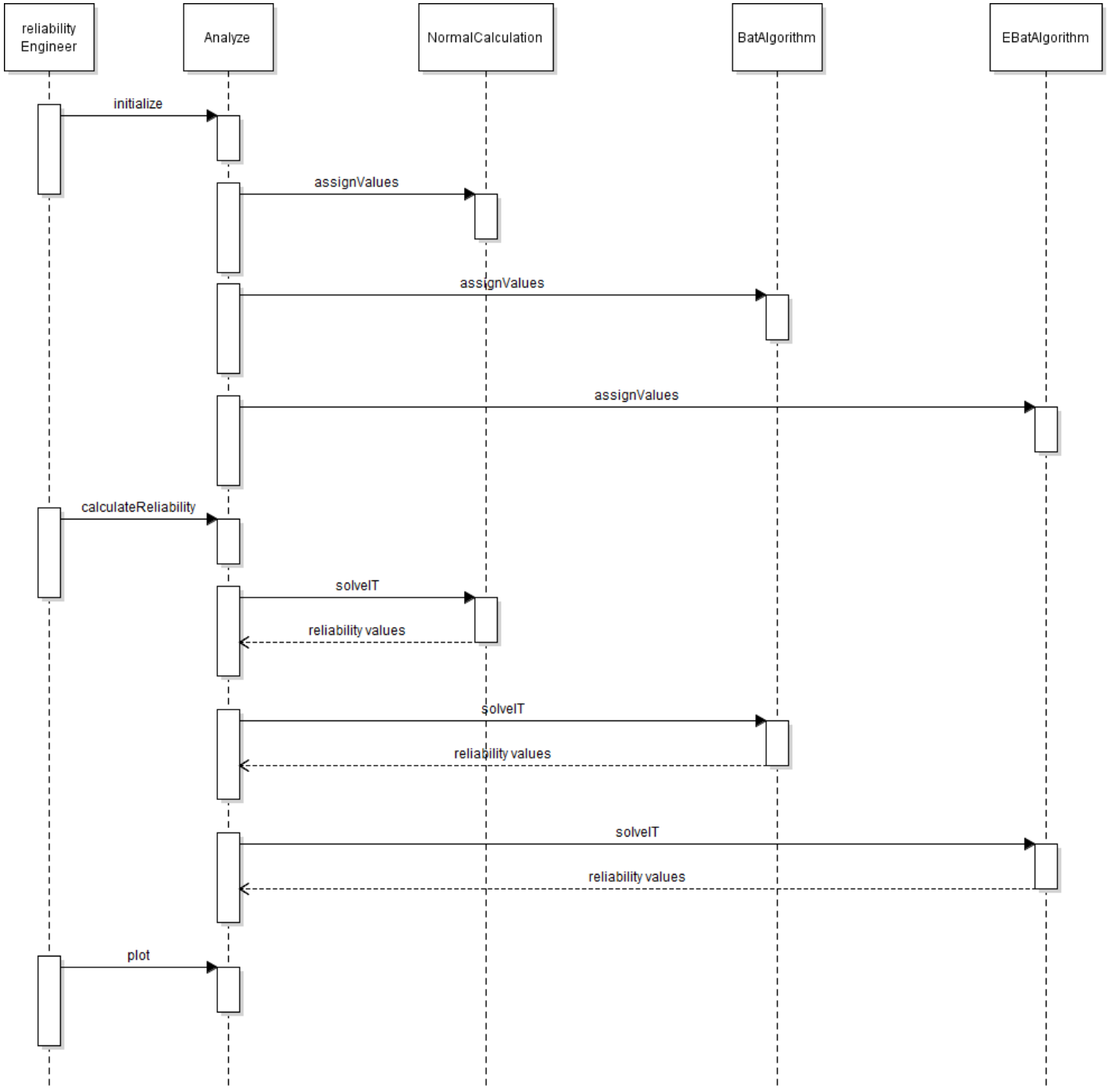
Use case Diagram:



Class Diagram:



Sequence Diagram:



SAMPLE SOURCE CODE:

Analyse

```
format long

inp = csvread('musa.csv');

n = size(inp);

n = n(1);

t = inp(:,2);

tn = max(t);

ti = sum(t);

x = 15;

display('Traditional calculation: ');

nbeta = normalcalculation(n,t,x);


integT= 0:10:1000;

upto = size(integT);

for i=1:upto(2)

        rel(i) = reliabilitycalculation(nbeta(1),nbeta(2),integT(i));

end

plot(integT,rel,'r');

xlabel('time');

ylabel('reliability');

title('Plot of Reliability Calculation');
```

```
display('##################################');

display('Solving using BAT Algorithm');
bbeta = batToSolveBETA1(n,t,x)
integT= 0:10:1000;
upto = size(integT);
for i=1:upto(2)
  rel1(i) = reliabilitycalculation(bbeta(1),bbeta(2),integT(i));
end
hold on
plot(integT,rel1,'b');
xlabel('time');
ylabel('reliability');
title('Plot of Reliability Calculation');
display('##################################');
display('Solving using E-BAT Algorithm');
bbeta = batToSolveBETA2(n,t,x)
integT= 0:10:1000;
upto = size(integT);
        for i=1:upto(2)
                rel2(i) = reliabilitycalculation(bbeta(1),bbeta(2),integT(i));
        end
hold on
```

```
plot(integT,rel2,'g');

xlabel('time');

ylabel('reliability');

title('Plot of Reliability Calculation');

format long

result = [integT;rel;rel1;rel2;rel-rel1;rel1-rel];

csvwrite('result.csv',result.');

display('#########NRMSE calculation###############')

nrmse = sqrt(sum(square(rel1 - rel))/sum(square(rel1)));

display('BAt vs traditional')

display(abs(nrmse))

nrmse = sqrt(sum(square(rel2 - rel))/sum(square(rel2)));

display('EBAt vs traditional')

display(abs(nrmse))

nrmse = sqrt(sum(square(rel2 - rel1))/sum(square(rel2)));

display('EBAt vs BAt')

display(abs(nrmse))
```

BAT_TO_SOLVE_BETA1

```
function beta = batToSolveBETA1(an,at,ax)

        mn=an;

         mt = at;

        mtn=max(at);
```

```
mx = ax;

mti = sum(mt);

para=[40 800 0.5 0.5];

n=para(1);

N_gen=para(2);

A=para(3);

r=para(4);

Qmin=0;

Qmax=2;

N_iter=0;

d=1;

Lb=-2*ones(1,d);

Ub=2*ones(1,d);

Q=zeros(n,1);
v=zeros(n,d);

        for i=1:n,

                Sol(i,:)=Lb+(Ub-Lb).*rand(1,d);

                Fitness(i)=Fun(Sol(i,:));

        end

 [fmin,I]=min(Fitness);

best=Sol(I,:);

        for t=1:N_gen,

                for i=1:n,
```

```
                    Q(i)=Qmin+(Qmin-Qmax)*rand;
                    v(i,:)=v(i,:)+(Sol(i,:)-best)*Q(i);
                    S(i,:)=Sol(i,:)+v(i,:);
                    Sol(i,:)=simplebounds(Sol(i,:),Lb,Ub);
                    if rand>r
                        S(i,:)=best+0.001*randn(1,d);
                    end
                Fnew=Fun(S(i,:));

                if (Fnew<=Fitness(i)) & (rand<A) ,
                Sol(i,:)=S(i,:);
                Fitness(i)=Fnew;
                end
                if Fnew<=fmin,
                best=S(i,:);
                fmin=Fnew;
                end
                end
                N_iter=N_iter+n;
end
beta(1) = best;
beta(2) = BetaZero(best);
```

```
function s=simplebounds(s,Lb,Ub)

    ns_tmp=s;

    I=ns_tmp<Lb;

    ns_tmp(I)=Lb(I);

    J=ns_tmp>Ub;

    ns_tmp(J)=Ub(J);

    s=ns_tmp;

end

function z=Fun(var)

    z = mn/var - (mn*(mtn+mx))/(exp(var*(mtn+mx))-1) - mti;

    z = abs(z);

end

function F = BetaZero(betaOne)

    F = mn/(1-exp(betaOne*-1*(mtn+mx)));

end
end
```

BAT TO SOLVE BETA2

```
function beta = batToSolveBETA2(an,at,ax)

    mn=an ;

    mt = at;

    mtn=max(at);

    mx = ax;

    mti = sum(mt);
```

```
para=[20 800 0.5 0.5];

n=para(1);

N_gen=para(2);

A=para(3);

r=para(4);

Qmin=0;

Qmax=2;

N_iter=0;

d=1;

Lb=-2*ones(1,d);

Ub=2*ones(1,d);

Q=zeros(n,1);

v=zeros(n,d);

        for i=1:n,

                Sol(i,:)=Lb+(Ub-Lb).*rand(1,d);

                Fitness(i)=Fun(Sol(i,:));

        end


 [fmin,I]=min(Fitness);

best=Sol(I,:);


w=rand(1,1);
l1 = 0.6;

l2 = 1-l1;
```

```
k = ceil(rand(1,1)*n);

        for t=1:N_gen,

                for i=1:n,

                Q(i)=Qmin+(Qmin-Qmax)*rand;

                v(i,:)=w*v(i,:)+(Sol(i,:)-best)*Q(i)*l1+(Sol(i,:)-Sol(k,:))*Q(i)*l2;


                S(i,:)=Sol(i,:)+v(i,:);

                Sol(i,:)=simplebounds(Sol(i,:),Lb,Ub);

                        if rand>r
                                S(i,:)=best+0.001*randn(1,d);

                        end

                Fnew=Fun(S(i,:));

                        if (Fnew<=Fitness(i)) & (rand<A) ,

                                Sol(i,:)=S(i,:);

                                Fitness(i)=Fnew;

                        end

                        if Fnew<=fmin,

                                best=S(i,:);

                                fmin=Fnew;

                        end
                end

                N_iter=N_iter+n;
        end
        beta(1) = best;
```

```
        beta(2) = BetaZero(best);


            function s=simplebounds(s,Lb,Ub)
                ns_tmp=s;

                I=ns_tmp<Lb;

                ns_tmp(I)=Lb(I);

                J=ns_tmp>Ub;

                ns_tmp(J)=Ub(J);
                s=ns_tmp;

            end


    function z=Fun(var)

        z = mn/var - (mn*(mtn+mx))/(exp(var*(mtn+mx))-1) - mti;

        z = abs(z);

    end

    function F = BetaZero(betaOne)

        F = mn/(1-exp(betaOne*-1*(mtn+mx)));

    end

end
```

NORMAL CALCUTAION

```
function answer = normalcalculation(an,at,ax)

    n = 0;

    t = 0;

    tn = 0;
```

```
x = 0;
ti = 0;

function  assignvalues

        n = an;

        t = at;
        tn= max(at);
        x = ax;
        ti = sum(t);

        display('normalcalculation: values are assignied')
end

function F =  likeLIhood(b0,b1)

        te1 = b0^n * b1^n;

        te2=0;

                for i= 1:n

                        te2 = te2+ exp(-b1*t(i));
                end

        te3 = exp(-b0*(1-exp(-b1*(tn+x))));

        F = te1*te2*te3;

end

function F = BetaZero(betaOne)

        F = n/(1-exp(betaOne*-1*(tn+x)));

end


function F = BetaOne(var)

        F = n/var - (n*(tn+x))/(exp(var*(tn+x))-1) - ti;
end
```

```
        function solveIT

                display('normalcalculation: solving the values')

                x1 = 1;

                [x1,fval]= fsolve(@BetaOne,x1);

                beta1 = x1

                beta0 = BetaZero(beta1)


                answer(1) = beta1;

                answer(2) = beta0;


          end

        assignvalues()
        solveIT()

end
```

RELIABILITY_CALC

```
function rel = reliabilitycalculation(b1,b0,at)

        myfun = @(x) b0*b1*exp(-b1*x);

        area=integral(myfun,at,at+5);

        rel = exp(-area);
end
```

LAG CLACULATION

```
t=csvread('musa.csv');

xlag = lagmatrix(t(:,2),[0 1 2 3 4 5]);
```

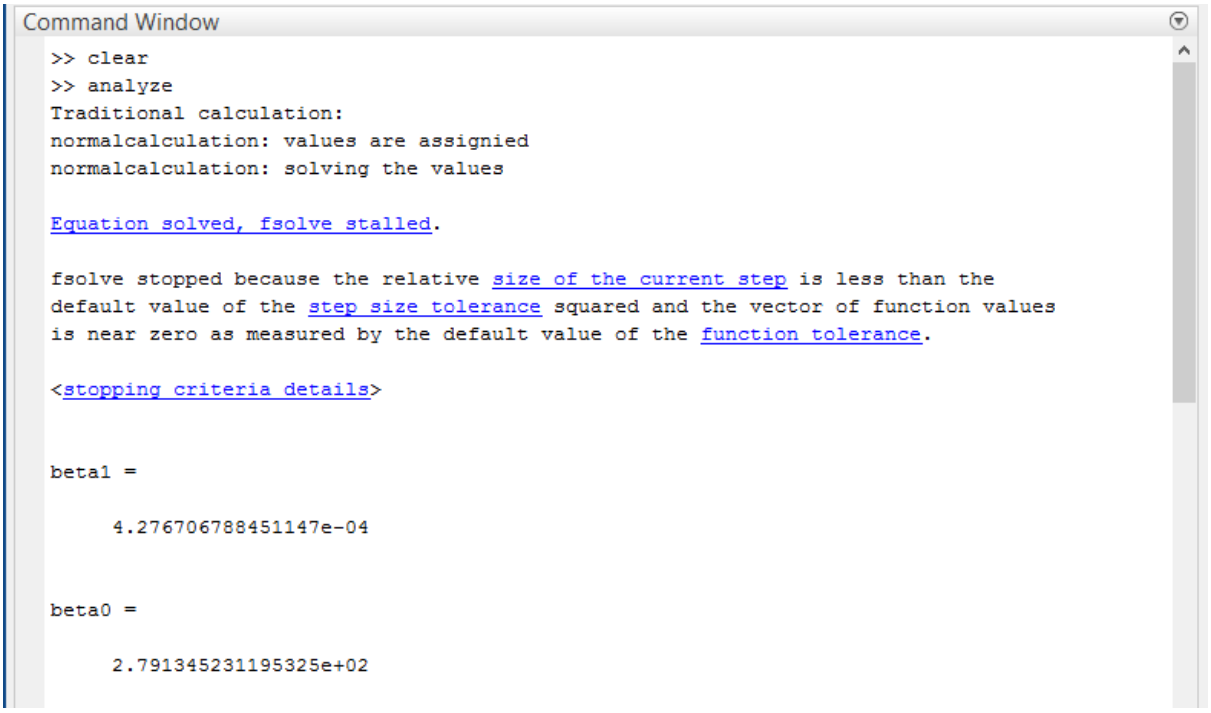
```
xlag = xlag(6:101,:);

xlag = [xlag(:,6) xlag(:,5) xlag(:,4) xlag(:,3) xlag(:,2) xlag(:,1) ];

csvwrite('lagcheck.csv',xlag);


nrmse=0;

nrmse1=0;

for lop = 1:5

format long

inp = xlag(:,lop);

n = size(inp);

n = n(1);

t = inp;

tn = max(t);

ti = sum(t);

x = 15;

display('Traditional calculation: ');

display('goes to normal calculation');

nbeta = normalcalculation(n,t,x);

 integT= 0:10:1000;

upto = size(integT);

for i=1:upto(2)
            rel(i) = reliabilitycalculation(nbeta(1),nbeta(2),integT(i));
end
```

```
plot(integT,rel,'r');

xlabel('time');

ylabel('reliability');

title('Plot of Reliability Calculation');

 display('##################################');

 display('Solving using BAT Algorithm');

bbeta = batToSolveBETA1(n,t,x)

integT= 0:10:1000;

upto = size(integT);

for i=1:upto(2)
        rel1(i) = reliabilitycalculation(bbeta(1),bbeta(2),integT(i));
end

hold on

%subplot(2,1,2);
plot(integT,rel1,'b');

%legend('normal','bat');
xlabel('time');

ylabel('reliability');

title('Plot of Reliability Calculation');

 display('##################################');

 display('Solving using E-BAT Algorithm');

bbeta = batToSolveBETA2(n,t,x)

integT= 0:10:1000;

upto = size(integT);
```

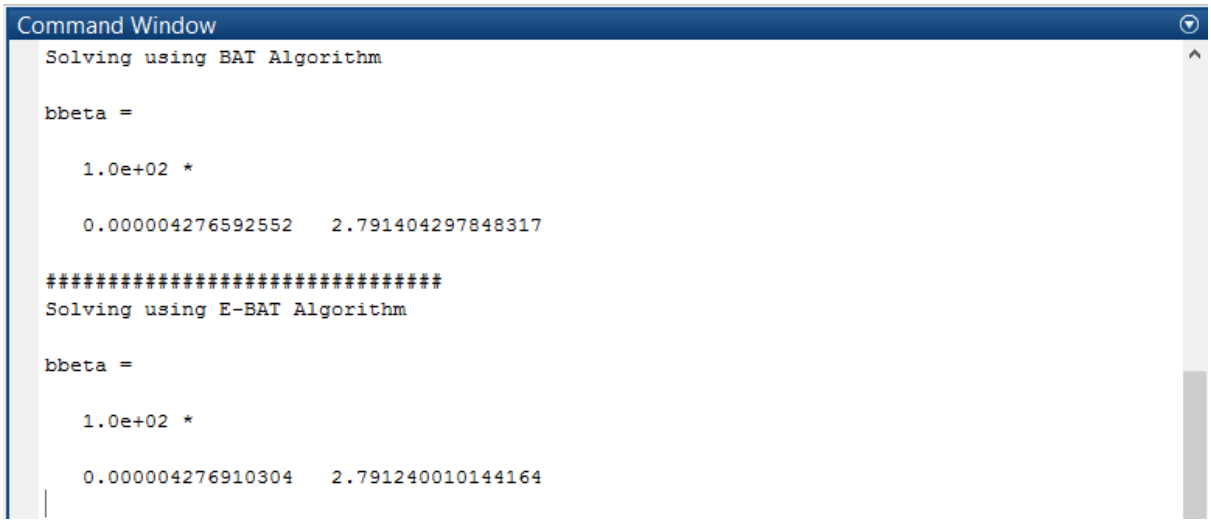
```
for i=1:upto(2)
        rel2(i) = reliabilitycalculation(bbeta(1),bbeta(2),integT(i));
end

hold on

plot(integT,rel2,'g');

xlabel('time');

ylabel('reliability');

title('Plot of Reliability Calculation');

format long

result = [integT;rel;rel1;rel2];

csvwrite(strcat('result',int2str(lop),'.csv'),result.');

display(strcat('dumped into:  result',int2str(lop),'.csv'));

display('#########NRMSE calculation##############')

nrmse(lop) = abs(sqrt(sum(square(rel - rel2))/sum(square(rel))));

display('EBAt vs tradi')

display(abs(nrmse))

nrmse1(lop) = abs(sqrt(sum(square(rel - rel1))/sum(square(rel))));

display('Bat vs tradi')

display(abs(nrmse1))

display('>>>>>>>>>>>>>&&&&&&&&&&&&&&<<<<<<<<<<<<<<<<<<<')

end
```

SAMPLE OUTPUT:

Traditional method



BAT and EBAT method



Reliability Values

Command Window

```
modification: NormalRel : BatRel : EBatRel

ans =

   1.0e+03 *

                  0   0.000550873251103   0.000550885319228   0.000550887600133
   0.010000000000000   0.000552276757037   0.000552288555081   0.000552290784940
   0.020000000000000   0.000553677826363   0.000553689355099   0.000553691534060
   0.030000000000000   0.000555076448156   0.000555087708366   0.000555089836575
   0.040000000000000   0.000556472611629   0.000556483604099   0.000556485681705
   0.050000000000000   0.000557866306128   0.000557877031652   0.000557879058806
   0.060000000000000   0.000559257521138   0.000559267980517   0.000559269957369
   0.070000000000000   0.000560646246275   0.000560656440317   0.000560658367020
   0.080000000000000   0.000562032471294   0.000562042400811   0.000562044277520
   0.090000000000000   0.000563416186081   0.000563425851895   0.000563427678763
   0.100000000000000   0.000564797380657   0.000564806783593   0.000564808560777
   0.110000000000000   0.000566176045175   0.000566185186066   0.000566186913724
   0.120000000000000   0.000567552169922   0.000567561049606   0.000567562727896
   0.130000000000000   0.000568925745315   0.000568934364637   0.000568935993719
   0.140000000000000   0.000570296761903   0.000570305121715   0.000570306701748
   0.150000000000000   0.000571665210368   0.000571673311524   0.000571674842672
   0.160000000000000   0.000573031081519   0.000573038924882   0.000573040407306
   0.170000000000000   0.000574394366296   0.000574401952733   0.000574403386599
   0.180000000000000   0.000575755055769   0.000575762386154   0.000575763771625
   0.190000000000000   0.000577113141137   0.000577120216346   0.000577121553589
   0.200000000000000   0.000578468613723   0.000578475434642   0.000578476723824
   0.210000000000000   0.000579821464983   0.000579828032500   0.000579829273789
```

Plot

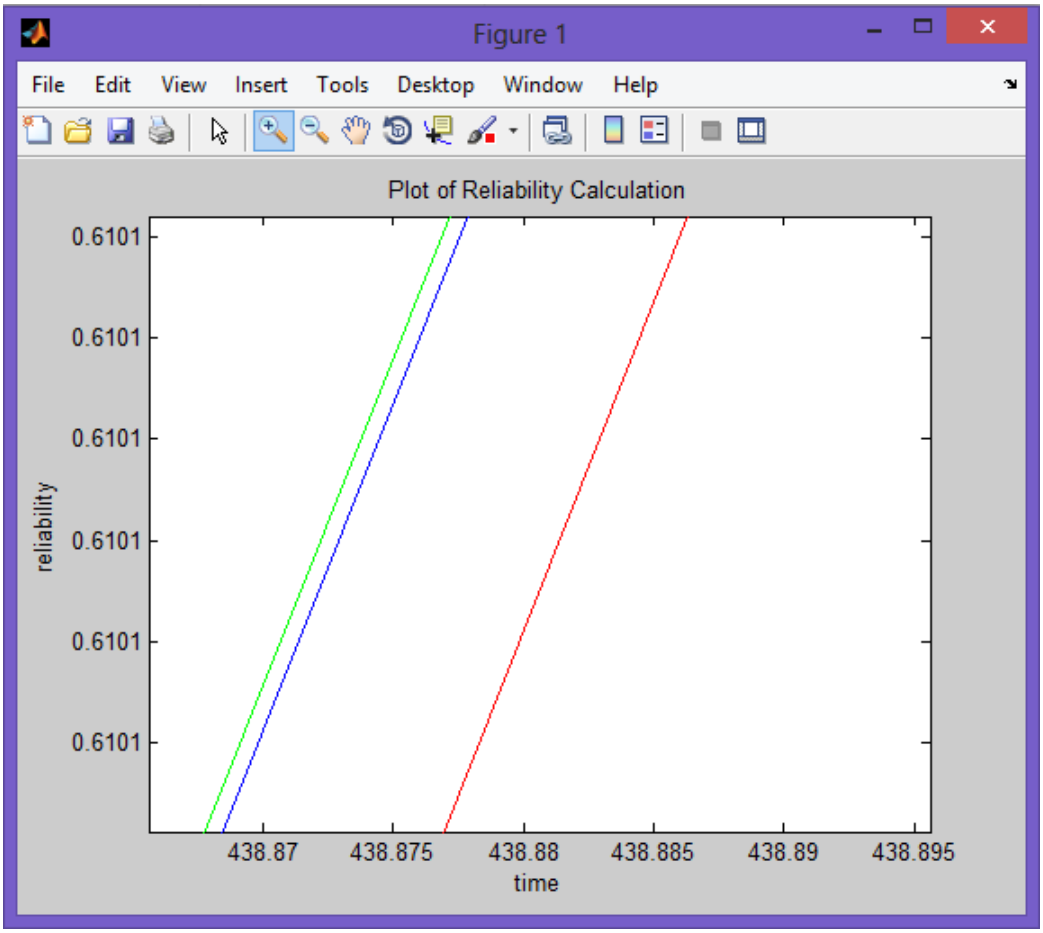


Figure 1
File Edit View Insert Tools Desktop Window Help
Plot of Reliability Calculation
0.6101
0.6101
0.6101
0.6101
0.6101
0.6101
reliability
438.87
438.875
438.88
438.885
438.89
438.895
time

CONCLUSION:

In this project, we obtain the optimised value of reliability through Enhance BAT (EBAT) algorithm. We see that the reliability calculated by traditional method is not optimal for complex problems. BAT is bad at Exploration and Exploitation. This problem has been resolved by introducing two modifications to the original BAT algorithm. The two modifications are i) Inertia weight modification ii) Distribution of population modification.

In this project, we compare the reliability calculated by employing traditional method, BAT algorithm and Enhanced BAT algorithm. The EBAT guaranteed better reliability compared to traditional method and BAT algorithm. The BAT has poor local and global search characteristics. The results show that the EBAT algorithm is superior to traditional and BAT algorithm.

BIBLIOGRAPHY:

- E. Ali, Optimization of power system stabilizers using {BAT} search algorithm;

  Int. J. Electr. Power Energy Syst. 61 (2014) 681-000.

- Hasancebi, T. Teke, 0. Pekcan, A bat-inspired algorithmfor staructural optimization, Comput. Strad. 128 (2013) 77-90
- 0, Hasancebiebi, S. Carbas, Bat Inspired algorithm for discrete size optimization of steel frames, Adv. Eng. Softw. 67 (2014) 173-185
- R.K. Mohanty, V. Ravi, M.R. Patra, Software reliability prediction using group method of data handling, in: international Conferences on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC, 2009), LNA 15908, IIT New Delhi, 2009, pp. 344-351.
- R.K. Mohanty, V. Ravi, M.R. Patra, Software reliability prediction using genetic programming, in: The international Conferences of the Biologically Inspired Computing and Applications (BICA-2009), Bhubaneswar, India, 200Q, pp. 331-336.
- R.K. Mohanty, V. Ravi, M.R. Patra, The application of intelligent and soft-computing techniques to software engineering problems: a review, international Journal a Information and Decision Sciences 2 (3) (2010) 232-272.
- J.D. Musa, software reliability data, IEEE Computer Society Repository (1070).