System reliability bounds using LP

Nophi Ian D. Biton

June 24, 2023

1 Introduction

In this homework, convex optimization is performed using CVX in Matlab implementation `http://cvxr.com/cvx/`. The relevant files were downloaded first and the installation instructions were followed to set-up the `cvx` in the local computer.

There are two sets of example that were used here to demonstrate how to use the `cvx` platform to perform convex optimization. Structural system reliability analysis which is formulated as linear programming optimization problem [1]. Two examples will be solved in this category, namely, (i) Truss as a Series system and (ii) Daniels' Parallel system. The following results to this problem can be validated from the study of [1].

Problem 1: System reliability bounds by Linear Programming

The question is asked: what is the probability of failure $P_f = P(g(X) < 0)$ where $g(X)$ is the limit state function. The simplest expression of limit state function is $g(X) = R - S$ where R is the capacity and S is the demand. In structural reliability, there are three states that are describe:

$$g(X) = R - S \begin{cases} > 0 & \text{Safe State} \\ = 0 & \text{Limit State} \\ < 0 & \text{Failure State} \end{cases} \tag{1}$$

Both R, S can be defined as random variables with corresponding probability density function. The uncertainty concerning the demands or load effects S can be associated to earthquake, wind loads or even live loads and dead loads while the uncertainty in the capacity or resistance R can be associated to concrete strength, yield strength, modulus of elasticity, geometry of the structural members etc.

A structure is usually a complex system with different components involved. Thus, there can be various modes the structure may fail. A reliability of a single component in a structural system may not offer a comprehensive description of its overall failure. The more applicable method of quantifying the probability of failure of a structural system will be through system reliability analysis. The probability of failure of system can be quantified based on several component events. Mathematically it is written as

$$P_f = P\left(\bigcup_{k=1}^{K} \bigcap_{i \in C_k} (g_i(X) \leq 0)\right) \tag{2}$$

A system event is described by each component events. If the failure of the system occurs when one of its components failes, then it is a **Series system**. When the failure of system is described when all the component events fail, then it is called a **Parallel system**. Any other event not desribed as either series or parallel is called a **General system**. A component event may mean failure of different structural members, different load cases, various failure modes (yielding, crushing, fatigue etc.) etc. In structural design, there are different limit states that are considered such as strength limit states, serviceability limit state, fatigue limit state etc. When all the limit states are considered then it becomes a system reliability problem.

A brief description on how to calculate the bounds of the system failure probability in Eq. 2 using linear programming (LP) will be provided here. For a more detailed and complete explanation, the readers are referred to the study of [1].

Firstly, the sample space of the component events are subdivided into 2^n mutually exclusive and collectively exhaustive events (MECE). All these basic MECE events can be defined by a unique intersection of the component events E_i and its complementary $\overline{E}_i$. As shown in Fig. 1, 8 basic MECE events (denoted as e_i) divides the sample space for a system with 3 components. Let $p_i = P(e_i), i = 1, 2 \ldots 2^n$ to denote the probabilities of the basic MECE events.

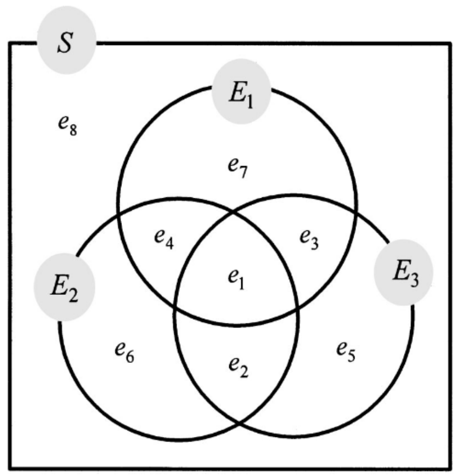

Figure 1: Mutually exclusive and completely exhaustive events [1]

The first set of the linear constraints for the LP problem is from the axioms of probability theory, which are

$$\sum_{i=1}^{2^n} p_i = 1 \tag{3}$$

$$p_i \geq 0, \forall i \tag{4}$$

Since the basic MECE events are defined such that it does not intersect from any other events, probability of any subset made of these events is simply the sum of the corresponding probabilities. In Fig. 1, the probability of event E_1 can be expressed as the sum of the basic MECE events

$$P(E_1) = P_1 = p_1 + p_3 + p_4 + p_7 \tag{5}$$

Additionally, the probability of the any intersection of the component events is given as the sum of the probabilities of the basic MECE events that constitute the intersection event. For example, the intersection of events E_1 and E_2 can be described as

$$P(E_1 E_2) = P_{12} = p_1 + p_4 \tag{6}$$

In general, these uni-, bi- and sometimes tri-component probabilities can be computed or given. In general, these expressions can be written as

$$P(E_i) = P_i = \sum_{r: e_r \subseteq E_i} p_r \tag{7}$$

$$P(E_i E_j) = P_{ij} = \sum_{r: e_r \subseteq E_i E_j} p_r \tag{8}$$

Given the probabilities $\mathbf{p} = (p_1, p_2 \ldots, p_{2^n})$, it can be seen that Eq. 5-8 are linear equality constraints of the form $\mathbf{a}^T \mathbf{p} = b$ where $\mathbf{a}$ is vector composed of 0's and 1's and b is the known uni-, bi- etc. component probabilities ($P(E_1)$ or $P(E_1 E_2)$ etc.).

Any Boolean function of the component events can also be considered as being composed of a subset of the basic MECE events. Thus, any system event E_{system} can be expressed in the form $P(E_{\text{system}}) = \mathbf{c}^T\mathbf{p}$ where $\mathbf{c}$ is vector composed of 0's and 1's. A systematic way of determining the 'event vector' $\mathbf{c}$ of the component events E_i is given in [2]. The lower bound of the system failure probability is defined as an LP

$$\begin{aligned} \min_{\mathbf{p}} \quad & P(E_{\text{system}}) = \mathbf{c}^T\mathbf{p} \\ \text{subject to} \quad & \sum_{i=1}^{2^n} p_i = 1 \\ & p_i \geq 0, \forall i \\ & P(E_i) = P_i = \sum_{r:e_r \subseteq E_i} p_r \\ & P(E_iE_j) = P_{ij} = \sum_{r:e_r \subseteq E_iE_j} p_r \end{aligned} \tag{9}$$

and the upper bound will be

$$\begin{aligned} \max_{\mathbf{p}} \quad & P(E_{\text{system}}) = \mathbf{c}^T\mathbf{p} \\ \text{subject to} \quad & \sum_{i=1}^{2^n} p_i = 1 \\ & p_i \geq 0, \forall i \\ & P(E_i) = P_i = \sum_{r:e_r \subseteq E_i} p_r \\ & P(E_iE_j) = P_{ij} = \sum_{r:e_r \subseteq E_iE_j} p_r \end{aligned} \tag{10}$$

The following examples described below follow the same LP problem formulation in Eq. 9 and 10. These problems were adapted from [1]. All the problems were solved using `cvx` platform.

1.1 Truss as a Series system

Consider as structural truss system shown in Fig. 2. Since this truss is statically determinate, the failure of one member will result to a failure of the entire structure. Thus, the system event for the failure of this truss can be defined as a series.

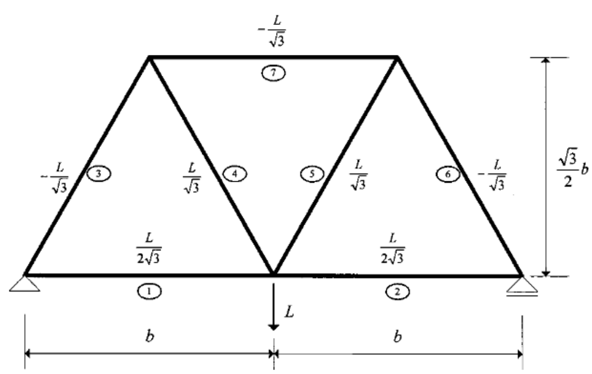

Figure 2: Statically determinate truss [1]

Let $X_i, i = 1, 2, \ldots 7$ be the strengths of the ith member. The member component will fail when the strength X_i for the ith member is exceeded by the internal force for each member due to the applied load

L. Thus, the failure events of the individual members are defined as $E_i = \{X_i \leq L/(2\sqrt{3})\}$ for $i = 1, 2$ and $E_i = \{X_i \leq L/(1\sqrt{3})\}$ for $i = 3, 4 \ldots 7$. Assume the deterministic value for $L = 100$ and the random distribution for $X_1 - X_2 \sim N(100, 20)$ and $X_3 - X_7 \sim N(200, 40)$ which are also assumed to jointly normal. Under these condition, the probabilities of the each component events are

$$P_i = P(E_i) = \Phi\left(\frac{100/\sqrt{3} - 200}{40}\right) = 1.88 \times 10^{-4}, \quad i = 1, 2 \ldots 7 \tag{11}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function. Additionally, the m-component probabilities is given by [1] :

$$P_{12\ldots m} = \Phi_m(u_1, u_2 \ldots u_m; \mathbf{R}) = \int_{-\infty}^{\infty}\left[\phi(t)\Pi_{i=1}^{m}\Phi\left(\frac{u_i - r_i t}{\sqrt{1 - r_i^2}}\right)\right] dt \tag{12}$$

where $\Phi_m(u_1, u_2 \ldots u_m; \mathbf{R})$ is the m-variate standard normal CDF with correlation matrix $\mathbf{R} = [\rho_{ij}]$ at coordinates $u_i = (100/\sqrt(3) - 200)/40$ and $\phi(\cdot)$ denotes the one dimensional standard normal probability density function. For this example, the value of the correlation coefficients were set as $r_1 = 0.90, r_2 = 0.96, r_3 = 0.91, r_4 = 0.95, r_5 = 0.92, r_6 = 0.94$ and $r_7 = 0.93$. The Matlab implementation is shown below.

```
clc, clear; format compact;

n_events = 7; % number of component events

% defined the correlation coefficient
r = [0.90, 0.96, 0.91, 0.95, 0.92, 0.94, 0.93];
R = r'*r; for i=1:size(R,1), R(i,i)=1; end

% calculate the unicomponent probability in Eq. 11
ui = (100/sqrt(3)-200)/40;
P_Ei = mvncdf (ui,0);

% define the event matrix by zeros an ones
[C] = event_matrix(n_events); % procedure is defined by Kang 2012
n_mece = 2^n_events;

% Define the problem data
c_sys = [ones(n_mece-1,1);0]; % define the event vector for a series event

% Solve the problem using CVX for the lower bound
cvx_begin
    variable p(n_mece)
    minimize(c_sys' * p)
    subject to

        sum(p) == 1
        for i=1:n_mece
            p(i) >= 0
        end

        for i=1:n_events
            C(:,i)'*p==P_Ei
        end

cvx_end

LowerBound = cvx_optval

% Solve the problem using CVX for the upper bound
cvx_begin
    variable p(n_mece)
    maximize(c_sys' * p)
    subject to

        sum(p) == 1
        for i=1:n_mece
            p(i) >= 0
        end
```

```
        for i=1:n_events
            C(:,i)'*p==P_Ei
        end

cvx_end

UpperBound = cvx_optval
```

An additional constraint is used when the bi-component probabilities are used. The matlab implementation is given below:

```
clc, clear; format compact;

n_events = 7; % number of component events

% defined the correlation coefficient
r = [0.90, 0.96, 0.91, 0.95, 0.92, 0.94, 0.93];
R = r'*r; for i=1:size(R,1), R(i,i)=1; end

% calculate the unicomponent probability in Eq. 11
ui = (100/sqrt(3)-200)/40;
P_Ei = mvncdf (ui,0);

% calculate the unicomponent probability in Eq. 12
EiEj = nchoosek(1:7,2);
P_EiEj = zeros(1,size(EiEj,1));
for k=1:size(EiEj,1)
    P_EiEj(k)= mvncdf (ui*ones(1,2),zeros(1,2),R(EiEj(k,:),EiEj(k,:)));
end

% define the event matrix by zeros an ones
[C] = event_matrix(n_events); % procedure is defined by Kang 2012
n_mece = 2^n_events;

% Define the problem data
c_sys = [ones(n_mece-1,1);0]; % define the event vector for a series event

% Solve the problem using CVX for the lower bound
cvx_begin
    variable p(n_mece)
    minimize(c_sys' * p)
    subject to

        sum(p) == 1
        for i=1:n_mece
            p(i) >= 0
        end

        for i=1:n_events
            C(:,i)'*p==P_Ei
        end

        for i=1:length(P_EiEj)
            (C(:,EiEj(i,1)).*C(:,EiEj(i,2)))'*p==P_EiEj(i)
        end

cvx_end

LowerBound = cvx_optval

% Solve the problem using CVX for the upper bound
cvx_begin
    variable p(n_mece)
    maximize(c_sys' * p)
    subject to

        sum(p) == 1
        for i=1:n_mece
            p(i) >= 0
        end

```

```
        for i=1:n_events
            C(:,i)'*p==P_Ei
        end

        for i=1:length(P_EiEj)
            (C(:,EiEj(i,1)).*C(:,EiEj(i,2)))'*p==P_EiEj(i)
        end

cvx_end

UpperBound = cvx_optval
```

The summary of the results are provided below

Bounds ($\times 10^{-3}$)		Lower	Upper
Unicomponent	Song et. al., 2003	0.188	1.32
	cvx	0.18783	1.3148
Bicomponent	Song et. al., 2003	0.477	0.912
	cvx	0.4771	0.91216

1.2 Daniels' Parallel System

The description of this problem is stated in [1].This problem is a parallel system as opposed to the first problem which is a series. As a numerical example, a Daniels system with 6 wires is considered. The wire strengths are assumed to have the Weibull distribution with CDF: $F(x) = 1 - \exp(-\lambda x^{\beta}), 0 < x$, with $\lambda = 0.01$ and $\beta = 10$.

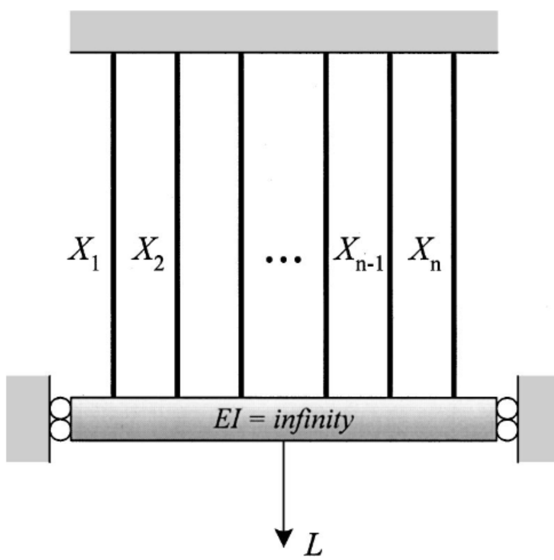

Figure 3: Daniels' parallel system [1]

```
clc, clear; format compact;
set(0,'defaultAxesFontSize',12)
set(0,'defaultTextFontName','Times New Roman')
set(0,'defaultAxesFontName','Times New Roman')


L_i = 5:0.1:8;
Pf_sys_exact=zeros(1,length(L_i));

```

```
f1 = figure;
set(f1,'units','inches','position',[1,1,4.5,3]);
leg_title = {};

for k=1:length(L_i)

    n = 6;
    L = L_i(k);

    x = zeros(n,1);
    for i=1:length(x), x(i) = L/(n-i+1); end

    lambda = 0.01;
    beta = 10;

    F = @(x,lambda,beta) 1-exp(-lambda*x.^(beta));
    b_n = F(x,lambda,beta);

    BB = zeros(n,n);
    for i=1:n
        bn = b_n(i);
        temp = [];
        for j=1:n-(i-1)
            temp = [temp,(bn^j)/(factorial(j))];
        end
        if i~=1
            BB(i,i-1:end) = [1,temp];
        else
            BB(1,1:end) = temp;
        end
        clear bn temp
    end
    Pf_sys_exact(k) = factorial(n) * det(BB);


    % calculate LP bounds
    [C] = event_matrix(n);
    n_mece = 2^n;

    P_Ei = zeros(1,length(x));
    for i=1:length(x)
        P_Ei(i) = Fi(x(i),i);
    end

    EiEj = nchoosek(1:n,2);
    for m=1:size(EiEj,1)
        P_EiEj(m)= Fij(x,EiEj(m,1),EiEj(m,2));
    end


    c_sys = [1;zeros(n_mece-1,1)];

    % Solve the problem using CVX
    cvx_begin
        variable p(n_mece)
        maximize(c_sys' * p)
        subject to
            sum(p) == 1

            for i=1:n_mece
                p(i) >= 0
            end

            for i=1:n
                C(:,i)'*p == P_Ei(i)
            end

    cvx_end

    LP_unicomp_UB(k) = cvx_optval;

    % Solve the problem using CVX
```

```
    cvx_begin
        variable p(n_mece)      % Define the optimization variable
        maximize(c_sys' * p)    % Define the objective function
        subject to
            sum(p) == 1

            for i=1:n_mece
                p(i) >= 0
            end

            for i=1:n
                C(:,i)'*p == P_Ei(i)
            end

            for i=1:length(P_EiEj)
                (C(:,EiEj(i,1)).*C(:,EiEj(i,2)))'*p == P_EiEj(i)
            end

    cvx_end

    LP_bicomp_UB(k) = cvx_optval;

end

semilogy(L_i,Pf_sys_exact,'k-',LineWidth=2.5); hold on; leg_title{1} = 'Exact';
semilogy(L_i,LP_unicomp_UB,'k-.'); hold on; leg_title{2} = 'Unicomponent (cvx)';
semilogy(L_i,LP_bicomp_UB,'k--'); hold on; leg_title{3} = 'Bicomponent (cvx)';

L_i = 5.7:0.1:8;

for k=1:length(L_i)

    n = 6;
    L = L_i(k);

    x = zeros(n,1);
    for i=1:length(x), x(i) = L/(n-i+1); end

    % calculate LP bounds
    [C] = event_matrix(n);
    n_mece = 2^n;

    P_Ei = zeros(1,length(x));
    for i=1:length(x)
        P_Ei(i) = Fi(x(i),i);
    end

    EiEj = nchoosek(1:n,2);
    for m=1:size(EiEj,1)
        P_EiEj(m)= Fij(x,EiEj(m,1),EiEj(m,2));
    end

    c_sys = [1;zeros(n_mece-1,1)];

    % Solve the problem using CVX
    cvx_begin
        variable p(n_mece)      % Define the optimization variable
        minimize(c_sys' * p)    % Define the objective function
        subject to
            sum(p) == 1

            for i=1:n_mece
                p(i) >= 0
            end

            for i=1:n
                C(:,i)'*p == P_Ei(i)
            end

            for i=1:length(P_EiEj)
                (C(:,EiEj(i,1)).*C(:,EiEj(i,2)))'*p == P_EiEj(i)
            end
```

```

    cvx_end

    LP_bicomp_LB(k) = cvx_optval;

end

semilogy(L_i,LP_bicomp_LB,'k--'); hold on; leg_title{4} = '';

L_i = 7.3:0.1:8;

for k=1:length(L_i)

    n = 6;
    L = L_i(k);

    x = zeros(n,1);
    for i=1:length(x), x(i) = L/(n-i+1); end

    % calculate LP bounds
    [C] = event_matrix(n);
    n_mece = 2^n;

    P_Ei = zeros(1,length(x));
    for i=1:length(x)
        P_Ei(i) = Fi(x(i),i);
    end

    EiEj = nchoosek(1:n,2);
    for m=1:size(EiEj,1)
        P_EiEj(m)= Fij(x,EiEj(m,1),EiEj(m,2));
    end

    c_sys = [1;zeros(n_mece-1,1)];

    % Solve the problem using CVX
    cvx_begin
        variable p(n_mece)
        minimize(c_sys' * p)
        subject to
            sum(p) == 1

            for i=1:n_mece
                p(i) >= 0
            end

            for i=1:n
                C(:,i)'*p == P_Ei(i)
            end

    cvx_end

    LP_unicomp_LB(k) = cvx_optval;

end

semilogy(L_i,LP_unicomp_LB,'k-.'); hold off; leg_title{5} = '';


xlabel('Load, L')
ylabel('Failure Probability, P_f')

legend(leg_title,Location="southeast");
filename = fullfile(strcat(pwd,'\Plots\','daniels_result.png'));
exportgraphics(gcf,filename,'Resolution',2000);
```

The results are shown below. These results closely resembles from the [1].

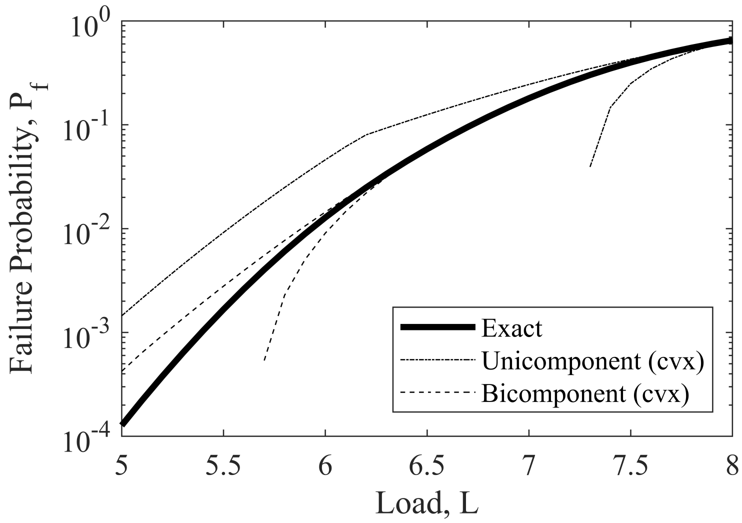

Figure 4: Daniels' parallel system failure probability bounds.

References

[1] J. Song and A. D. Kiureghian, "Bounds on system reliability by linear programming," *Journal of Engineering Mechanics*, vol. 129, no. 6, pp. 627–636, 2003.

[2] W.-H. Kang, Y.-J. Lee, J. Song, and B. Gencturk, "Further development of matrix-based system reliability method and applications to structural systems," *Structure and Infrastructure Engineering*, vol. 8, pp. 441–457, 5 2012. doi: 10.1080/15732479.2010.539060.