Priedai

A priedas. Santvaros optimizavimo programinis kodas

3.1 poskyrio, antrojo skaitinio eksperimento programinis kodas parašytas MATLAB programavimo terpėje

Problemos aprasymas

```
==========SANTVAROS TURIO MINIMIZACIJOS MODELIS ======================

 pagal matematini modeli:

         min L'A
 kai:   [Fi][G]{lambda}-sigmay[Gama]{A}<=[-Ne_max; Ne_min];        takumas

nt l    {lambda_cr_}.*(sigmay[G1]{A}-sigmay[G1](chi){A}) = 0   <<<NAUJA

        {lambda}'([Fi][G]{lambda}-sigmay[Gama]{A}+[Ne_max; -Ne_min])=0
t nl    [H]{lambda} <= u_sup - ue_max
        -lambda < 0
```

```
========================================================================

%}
clc; clear; close all;
tic
```

Geometrijos ivedimas (keiciamas blokas)

```
global n k sigmay Gama L Vchi bnq Z

elementu_mazgai = [ 1 2; 1 6; 1 7; 2 3; 2 6; 2 7; 2 8; 3 4; 3 7; 3 8; 3 9;
                    4 5; 4 8; 4 9; 4 10; 5 9; 5 10; 6 7; 7 8; 8 9; 9 10];

% koodinaciu pradzia -kairys apatinis konstrukcijos kampas.
% x -horizontaliai, y-vertikaliai aukstyn.
% klasikine formuluote alfa > 0 pirmam ketvirtyje.
mazgu_koordinates = [0 4; 3 4; 6 4; 9 4; 12 4; 0 0; 3 0.875; 6 1.5; 9
1.875; 12 2];
% Mazgu koordinates x y metrais
% globalus poslinkiai: po du kiekvienam mazgui is eiles
% pirmas mazgo poslinkis x - horizontalus i desine, antras y kryptimi
zemyn
atramos = [1 11 12];   % suvarzyti poslinkiai is visu galimu

mastelis = 15;     % mastelis kiek kartu padidinsime poslinkius brezinyje

% geometrijos ivedimo pabaiga

figure('units','normalized','outerposition',[0 0 1 1]); % fullscreen
subplot(2, 2, [1 2]);  % PRADINE SCHEMA
brezinys_santvarai(elementu_mazgai,mazgu_koordinates);
```

Pradiniai duomenys apie medziagas ir skerspjuvius

```
sigmay = 235e3; % kPa takumo itempis
E = 205e6;      % kPa Tamprumo modulis

% skirtingi skerspjuviu plotai. valcuoti SHS (kvadratiniai deziniai):

% A1 apatine juosta             [m^2]
```

```
% A2 virsutine juosta           [m^2]
% A3 statramsciai (vertikalus)  [m^2]
% A4 spyriai (istrizi)          [m^2]

A_min = [ 0.0006 0.0006 0.0006 0.0006  ]';  % minimalus plotai 50x50x3 mm
[m^2]
% Skerspjuviu pradines charakteristikos
d1 = 0.16; d2 = 0.14; d3 = 0.1; d4 = 0.1;      % Kvadratinio krastine [m]
t1 = 0.01; t2 = 0.01; t3 = 0.01; t4 = 0.01;  % sienutes storiai [m]

% Skerpsjuviu plotai. Bendra formule:
As = @(b,t) 2*b*t + 2*(b-2*t)*t - (((1.5*t)^2)*(4-pi)) + t^2*(4-pi);
% Keturiem skirtingiem strypam:
A1 = As(d1,t1); A2 = As(d2,t2); A3 = As(d3,t3); A4 = As(d4,t4);

% A1= 8.73e-4;A2=A1;A3=A1;A4=A1;

As = [A1; A2; A3; A4];   % bendras skerspjuviu plotu vektorius
k = size(As,1);          % skaicius kiek skirtingu skerspjuviu

% konfiguracijos matrica. Parodo kur kokie skerspjuviai
g1 = [ 0 1 0 0
       0 0 0 1
       0 0 1 0
       0 0 0 1 ];

Gama = [0 1 0 0
        0 0 1 0
        0 0 0 1
        g1;g1;g1;
        0 0 0 1
        0 0 1 0
        [ones(4,1) zeros(4,3)]];

% pradiniu duomenu bloko pabaiga
```

Tamprus skaiciavimai

```
% Pusiausvyros salygu koeficientu matrica:
[A] = A_matrica_santvarai(elementu_mazgai,mazgu_koordinates,atramos);
[m,n] = size(A);    % m-globalus posliniai, n-irazu skaicius
```

```
Alk = Gama*As;      % skerspjuviu plotu vektorius
elementu_skaicius = size(elementu_mazgai,1);  % Elementu skaicius
xx = mazgu_koordinates(:,1);      % mazgu koordinates x
yy = mazgu_koordinates(:,2);      % mazgu koordinates y
%  Nustatome elementu ilgius
L = zeros(1,n);
for elem = 1:elementu_skaicius;
    indice = elementu_mazgai(elem,:);
    xa = xx(indice(2))-xx(indice(1));
    ya = yy(indice(2))-yy(indice(1));
    elemento_ilgis = sqrt(xa*xa+ya*ya);
    L(elem) = elemento_ilgis;
end

% Judamos apkrovos pora
f = [ -140 -70 ]; % kN

F_nul = zeros(1,size(mazgu_koordinates,1)*2-size(atramos,2));
% Nuolatine apkrova:
F_nuol = F_nul; F_nuol([1 3 5 7 9])= -15;  F_nuol(17)= -0;  F_nuol(9)= -0;
F_nuol(8)= 0;
Mfk_nuol = repmat(F_nuol',1,4); % Apkrovu sumine matrica 14 16 18 20

F1 = F_nul; F1([1  3]) = f;
F2 = F_nul; F2([3  5]) = f;
F3 = F_nul; F3([5  7]) = f;
F4 = F_nul; F4([7  9]) = f;

MFk_judama = [F1' F2' F3' F4']; % Apkrovu sumine matrica
mf = size(MFk_judama,2);

MFk = MFk_judama*1.0 + Mfk_nuol*1.0;  % Charakteristiniu apkrovu matrica
% gama = 1.5;     % Dalinis koeficientas apkrovai
MFd = MFk_judama*1.3 + Mfk_nuol*1.35 ; % Skaiciuotiniu apkrovu matrica

Vchi = 0.9*ones(n,1); % Pradiniai klupumo koeficientai
```

ITERACINIS CIKLAS

```
iter = 20;                    % Iteraciju skaicius
x0 = [zeros(2*n,1); As];    % Pradines nezinomuju reikmes
```

```
for i = 1:iter;

% Konstrukcijos pasiduodamumo matrica:
% pagal elementu pasiduodamumo matricas dk=lk/(E*Ak):
D = (1/E)*eye(elementu_skaicius)*diag(L)*diag(Alk.^-1);

% Tamprus skaiciavimas:
alfa = D^-1*A'*(A/D*A')^-1; % Irazu influentine matrica
beta = ((A/D)*A')^-1;       % Poslinkiu influentine matrica

% Formuojam irazu matrica pagal visas apkrovas:
MNe = zeros(n,mf);
for j = 1:mf
    MNe(:,j) = alfa*MFd(:,j);
end

%Sudarome min ir max vektorius:
Ne_min = zeros(n,1);
Ne_max = zeros(n,1);
for ff=1:n
    Nmin=min(MNe(ff,:));
    if Nmin >= 0
        Ne_min(ff,1)=0;
    else
        Ne_min(ff,1)=Nmin;
    end

    Nmax=max(MNe(ff,:));
    if Nmax <= 0
        Ne_max(ff,1)=0;
    else
        Ne_max(ff,1)=Nmax;
    end
end

% Formuojam poslinkiu matrica pagal visas apkrovas:
MUe=zeros(m,mf);
for g = 1:mf
    MUe(:,g)=beta*MFk(:,g);
end
%Sudarome poslinkiu min ir max vektorius:
ue_min=zeros(m,1); ue_max=zeros(m,1);
for tu=1:m
```

```
    Umin=min(MUe(tu,:));
    if Umin >= 0
        ue_min(tu,1)=0;
    else
        ue_min(tu,1)=Umin;
    end

    Umax=max(MUe(tu,:));
    if Umax <= 0
        ue_max(tu,1)=0;
    else
        ue_max(tu,1)=Umax;
    end
end

% PAGRINDINIU LYGCIU SUDARYMAS
Fi = [ eye(n)
      -eye(n)]; % takumo salygu matrica

Hsubr = (A/D*A')^-1*A/D;    % liekamuju poslinkiu infl matrica
Gsubr = D\A'*Hsubr-inv(D);  % liekamuju irazu infl matrica
H = Hsubr*Fi'; G = Gsubr*Fi';

Aeq = []; beq = []; % tiesiniu apribojimu lygybemis nera

Anq = zeros(4*n+k+1,2*n+k);             % Apribojimai nelygybemis

bnq = zeros(4*n+k+1,1);                 % laivieji nariai

Anq(1:2*n,1:2*n) = Fi*G;                % takumas
Anq(1:2*n,2*n+1:2*n+k) = -sigmay*[Gama; Gama].*[ones(n,k);
repmat(Vchi,1,k)]; %tak
Anq(2*n+1:4*n,1:2*n) = -eye(2*n);       % -lambda <= 0
Anq(4*n+1:4*n+k,2*n+1:2*n+k) = -eye(k); % -A <= -A_min

% Kai ribojame du vertikalius poslinkius +/-
Anq(4*n+k+1:4*n+k+1,1:2*n) = [-H(17,:)]; % poslinkiai

bnq(1:2*n,1) = [-Ne_max; Ne_min];       % takumas
bnq(4*n+1:4*n+k,1) = -A_min;            % plotai konstruktyviai

% Kai ribojame du vertikalius poslinkius +/-
bnq(4*n+k+1:4*n+k+1,1) = [0.035 + ue_min(17)]; %
```

```
Z = Fi*G;           % matrica Z -grieztumo salygoms uzrasyti
lb = []; ub = [];   % Apatines ir virs kintamuju reiksmes
% x=[];
% san_funkcija = @(x)san_funkcija(x); %,n,k,Gama,L);
% san_apribojimai =
@(x,Cnq,Ceq)san_apribojimai(x,n,k,sigmay,Vchi,Gama,bnq,Z);


%==============OPTIMIZACIJOS PAPROGRAME====START==================
options = optimset('TolCon',1e-12,'TolFun',1e-12, 'TolX',1e-12, ...
    'Algorithm','active-set','MaxFunEvals',1e4,'Maxiter',1e4); %'trust-
region-reflective''active-set' 'interior-point'
[x,fval,exitquad,output,lambda]= fmincon(@san_funkcija,x0,Anq,bnq,...
    Aeq,beq,lb,ub,@san_apribojimai, options);
%===============OPTIMIZACIJOS PAPROGRAME====END==================

% PO-OPTIMIZACINIS KINTAMUJU APRASYMAS (KEITIMAS NAUJAIS)
x0 = x;

pl_daugikliai = x(1:2*n);

A_n = x(2*n+1:2*n+k);   % nauji skerpjuvio plotai
Alk = Gama*A_n;         % nauji skerpjuvio visai konstrukcijai

T(i,:) = [t1 t2 t3 t4];
% ieskom sieneles storiu is gautu skerspjuvio plotu
tb = @(d,Aa) (4*d - sqrt((4*d)^2-4*(9-1.25*pi)*Aa)) / (2*(9-1.25*pi));
t1 = real(tb(d1,A_n(1)));
t2 = real(tb(d2,A_n(2)));
t3 = real(tb(d3,A_n(3)));
t4 = real(tb(d4,A_n(4)));

% Inercijos momentu bendra formule. Cia supaprastinta su mathcadu.
% Tikroji yra mathcad faile...
I = @(b,t) ((79*b*t^3)/12) + ((2*b^3*t)/3) - ((83*t^4)/12) - ...
    ((13*b^2*t^2)/4) + ((133*pi*t^4)/64) - ((11*pi*b*t^3)/8) + ...
    ((5*pi*b^2*t^2)/16); % yra netikslumu??
% Inercijos momentai skirtingiems stypams:
I1 = I(d1,t1); I2 = I(d2,t2); I3 = I(d3,t3); I4 = I(d4,t4);
I_n = Gama*[I1;I2;I3;I4];
```

```
alfakoef = 0.49; % 0.49 - saltai formuoti (atrodo buna dazniau)
Vmiu = Gama * 0.9*[1 1 1 1]';
VLcr = L' .* Vmiu; % Elementu skaiciuojamuju ilgiu vektorius:

% Klupumo charakteristikos pagal EN:
VNcr = (pi^2*E*I_n)./(VLcr.^2); % Kritines jegos:
Vlambda = ((sigmay*Alk)./VNcr).^(1/2); % Salyginiai klupumo koef
Vfi = 0.5*(1+alfakoef*(Vlambda-0.2)+(Vlambda.^2)); % koef. fi
Vchi = 1./(Vfi+(((Vfi.^2)-(Vlambda.^2)).^(1/2))); % klup. koef

l = length(Vchi); % Ivykdom salyga, kad chi<=1:
 for j=1:l
    if Vchi(j,1)>1;
        Vchi(j,1)=1;
    else
        Vchi(j,1)=Vchi(j,1);
    end
 end
Vchi;


ur = H*x(1:2*n);
UMAX = ue_max+ur;
UMIN = ue_min+ur;

UMIN_17 = UMIN(17)

poslinkiai = UMIN;
subplot(2, 2, 3,'Align');
hold off
brezinys_santvarai_deformuotas(elementu_mazgai,mazgu_koordinates,...
    atramos,mastelis,poslinkiai,pl_daugikliai);


for tt = 1:k  %% REZULTATU MASYVO APRASYMAS
    rez(i,tt)=10000*x(2*n+tt);
    rez(i,k+1)=fval;
    rez(i,k+2)=exitquad;
end
hold off
subplot(2, 2, 4);
hold off
plot(rez(:,k+1));
```

```
pause(.01);
end
xlabel(['OPTIMIZACIJA BAIGTA'],'FontWeight','bold', 'FontSize',15, ...
    'Color','black');
```

REZULTATU SPAUSDINIMAS I EKRANA

```
disp(' '); disp(rez); % Rezultatai

Nr = G*x(1:2*n);
N_max = Ne_max+Nr;
N_min = Ne_min+Nr;
NO_max = sigmay*Alk;
NO_cr = sigmay*Alk.*Vchi;
Iraz =[Ne_max, Ne_min Nr, N_max N_min NO_max NO_cr];
numeriai = {'1','2','3','4','5','6','7','8','9','10','11','12',...
    '13','14','15','16','17','18','19','20','21'};
Irazos = dataset({Iraz 'Ne_max','Ne_min','Nr','N_max','N_min','NO_max',...
    'NO_cr'}, 'obsnames',  cellstr(numeriai));

disp(Irazos);


uu = [  ue_min    ue_max   ur      UMAX      UMIN];
pavv = {'1','2','3','4','5','6','7','8','9','10','11','12',...
    '13','14','15','16','17'};
Poslinkiai = dataset({uu 'ue_min','ue_max','u_r', 'u_max', 'u_min' }, ...
                'obsnames',  cellstr(pavv));
disp(Poslinkiai)

% T  % cia sieneliu storiai suvestiniai
% UMIN_17
toc
```

```
function [A] = A_matrica_santvarai(elementu_mazgai,mazgu_koordinates,...
    atramos)
```

```
%   failas sudarantis pusiausvyros lygciu koeficientu matrica santvarai

elementu_skaicius = size(elementu_mazgai,1);  % Elementu skaicius
mazgu_skaicius = size(mazgu_koordinates,1);   % Mazgu skaicius

xx = mazgu_koordinates(:,1);      % mazgu koordinates x
yy = mazgu_koordinates(:,2);      % mazgu koordinates y

LL = 2*mazgu_skaicius;            % Laisvumo laipsnis (bendras su atramom)

for k = 1:elementu_skaicius;
    indice = elementu_mazgai(k,:);
    % elemento galu globalus poslinkiai:
    elemento_LL = [ indice(1)*2-1, indice(1)*2, ...
                    indice(2)*2-1, indice(2)*2 ];

    xa = xx(indice(2))-xx(indice(1));
    ya = yy(indice(2))-yy(indice(1));
    elemento_ilgis = sqrt(xa*xa+ya*ya);
    cosa = xa/elemento_ilgis;
    sina = ya/elemento_ilgis;

    % k-ojo elemento transformacijos matricos pamatrice Tk:
    Tk = [cosa sina   0   0 ;
            0    0  cosa sina];

    % k-ojo elemento pusiausvyros lygciu lokali matrica Ak_strich
    Ak_strich = [ -1; 1 ];

    % Atskiro k-ojo elemento transformuota pusiausvyros pamatrice:
    Ak_be_c = Tk'*Ak_strich;

    % Visos konstrukcijos situ pagalbiniu Ak_be_c sujungtine matrica:
    A_subr(4*k-3:4*k,k) = Ak_be_c;


    [aktyvus_poslinkiai, ia] = setdiff(elemento_LL, atramos);
    % k-ojo elemento poslinkiu darnos matrica Ck
    Ck1 = zeros(4,LL);   % cia yra visi poslinkiai - kartu su atramom!!
        for j = 1:size(ia);
        Ck1(ia(j),aktyvus_poslinkiai(j)) = 1;
        end
```

```
    % Visi tariami konstrukcijos poslinkiai (su atraminiais):
    Visi_LL = (1:LL);
    % Tik galimi poslinkiai (atmetame itvirtinimus):
    Galimi_LL = setdiff(Visi_LL, atramos);

        for u = 1:size(Galimi_LL,2); % atmetam nereikalingas Ck eilutes
            Ck(:,u) = Ck1(:,Galimi_LL(u));
        end

    % Visos konstrukcijos darnos matrica C:
    C(4*k-3:4*k,:) = Ck;

end

% Konstrukcijos pusiausyros matrica:
    A = C'*A_subr;
end
```

```
function brezinys_santvarai(elementu_mazgai,mazgu_koordinates)
% Programele braizanti santvaras pagal koordinates

mazgu_skaicius = size(mazgu_koordinates,1);   % Mazgu skaicius
LL = 2*mazgu_skaicius;            % Laisvumo laipsnis (bendras su atramom)


elementu_skaicius = size(elementu_mazgai,1);  % Elementu skaicius
xx = mazgu_koordinates(:,1);                  % mazgu koordinates x
yy = mazgu_koordinates(:,2);                  % mazgu koordinates y


% Sukuriam kiekvieno elemento pradzios ir pabaigos mazgus atsirai
mazgas = elementu_mazgai';
for i = 1:2*elementu_skaicius
    elementuKoordinates(i,:) = mazgu_koordinates(mazgas(i),:);
end

% braizom pradine konstrukcijos schema:
for elem = 1:elementu_skaicius
    plot (elementuKoordinates(2*elem-1:2*elem,1),...
          elementuKoordinates(2*elem-1:2*elem,2),...
```

```
    '-ko','LineWidth',2,'MarkerSize',9,'MarkerFaceColor','w')
   hold on
end

axis equal
% axis ij   %  sitas padaro kitokias asis, kad is viraus zemyn eitu.
axis ([-2 max(max(xx))+2 min(yy)-3 max(max(yy))+3])
set(gca,'XTick',min(min(xx)):1:max(max(xx)))
set(gca,'YTick',min(min(yy)):1:max(max(yy)))
grid off

for j = 1:mazgu_skaicius

text(mazgu_koordinates(j,1),mazgu_koordinates(j,2),[num2str(j)],...
      'HorizontalAlignment','center','FontSize',6,...
      'margin',0.01);
end

txstr(1) = {'Konstrukcijos schema [matmenys metrais]',};
txstr(2) = {''};
title(txstr,'FontWeight','bold');

end
```

```
function brezinys_santvarai_deformuotas(elementu_mazgai, ...
    mazgu_koordinates,atramos,mastelis,poslinkiai,pl_daugikliai)
% Programele braizanti santvaras pagal koordinates

mazgu_skaicius = size(mazgu_koordinates,1);   % Mazgu skaicius
LL = 2*mazgu_skaicius;            % Laisvumo laipsnis (bendras su atramom)


elementu_skaicius = size(elementu_mazgai,1);  % Elementu skaicius
xx = mazgu_koordinates(:,1);                  % mazgu koordinates x
yy = mazgu_koordinates(:,2);                  % mazgu koordinates y

% Sukuriam kiekvieno elemento pradzios ir pabaigos mazgus atsirai
mazgas = elementu_mazgai';
for i = 1:2*elementu_skaicius
```

```
    elementuKoordinates(i,:) = mazgu_koordinates(mazgas(i),:);
end

% braizom pradine konstrukcijos schema:
for elem = 1:elementu_skaicius
    plot (elementuKoordinates(2*elem-1:2*elem,1),...
          elementuKoordinates(2*elem-1:2*elem,2),...
    '--k','LineWidth',1); %,'MarkerSize',9,'MarkerFaceColor','w')
   hold on
end

axis equal
% axis ij   %  sitas padaro kitokias asis, kad is viraus zemyn eitu.
axis ([-2 max(max(xx))+2 min(yy)-3 max(max(yy))+3])
set(gca,'XTick',min(min(xx)):max(max(xx)):max(max(xx)))
set(gca,'YTick',min(min(yy)):max(max(yy)):max(max(yy)))
grid off


    % Atkasinejam kokie yra kurie poslinkiai, nes turim tik atmestus:
    % Visi tariami konstrukcijos poslinkiai (su atraminiais):
    Visi_LL = (1:LL);
    % Tik galimi poslinkiai (atmetame itvirtinimus):
    Galimi_LL = setdiff(Visi_LL, atramos);
    posl = zeros(LL,1);
    for u = 1:size(Galimi_LL,2); % surenkam pilna posl vektoriu su atramom
        posl(Galimi_LL(u),1) = poslinkiai(u,1);
    end

        % sudedam teisingai mazgams poslinkiu matrica
    for p = 1:LL/2
        poslinkiai_mazgams(p,:) = [posl(2*p-1) posl(2*p)]; % ZENKLAS
BUVO!!
    end

% sukuriam mazgu koordinates su pridetu poslinkiu
naujosMazguKoord = mazgu_koordinates + mastelis*poslinkiai_mazgams;
for i = 1:2*elementu_skaicius
    deformuotosKoordinates(i,:) = naujosMazguKoord(mazgas(i),:);
end


% ziurim kurie strypai patyre plastines deformacijas
```

```
% ir isrenkame stypu kurie plastiskai deformavosi koordinates
tempimas = pl_daugikliai(1:elementu_skaicius);
gniuzdymas = pl_daugikliai(elementu_skaicius+1:2*elementu_skaicius);
 visi_tempiami = 0;  visi_gniuzdomi = 0;
for plast = 1:elementu_skaicius
    if tempimas(plast) >= 1e-6;
%         tempimas(plast)=1;
        visi_tempiami = visi_tempiami+1;
        tempiamo_koordinates(2*visi_tempiami-1:2*visi_tempiami,:)= ...
            deformuotosKoordinates(2*plast-1:2*plast,:);
%     else tempimas(plast) = 0;
    end
end
for plast_g = 1:elementu_skaicius
    if gniuzdymas(plast_g) >= 1e-6;
%         gniuzdymas(plast_g)=1;
        visi_gniuzdomi = visi_gniuzdomi+1;
        gniuzdomo_koordinates(2*visi_gniuzdomi-1:2*visi_gniuzdomi,:)= ...
            deformuotosKoordinates(2*plast_g-1:2*plast_g,:);
%     else gniuzdymas(plast_g) =0;
    end
end


hold on
% Braizome paprasta deformuota schema
for dk = 1:elementu_skaicius
    plot (deformuotosKoordinates(dk*2-
1:dk*2,1),deformuotosKoordinates(dk*2-1:dk*2,2),...
    '-k','LineWidth',2)
end


if visi_tempiami > 0
    for ttk = 1:size(tempiamo_koordinates,1)/2
    plot (tempiamo_koordinates(ttk*2-
1:ttk*2,1),tempiamo_koordinates(ttk*2-1:ttk*2,2),...
    '-r','LineWidth',2)
    end
end
if visi_gniuzdomi > 0
    for gk = 1:size(gniuzdomo_koordinates,1)/2
    plot (gniuzdomo_koordinates(gk*2-1:gk*2,1),gniuzdomo_koordinates(gk*2-
```

```
1:gk*2,2),...
    '-g','LineWidth',2)
    end
end

txstr(1) = {'Deformuota schema ir plastines deformacijos',};
txstr(2) = {''};
title(txstr,'FontWeight','bold');
xlabel(['Deformacijos yra padidintos ',num2str(mastelis),' kartu'])
% legend('pradine schema','deformuota schema',...
%     'Location','SouthOutside') % Brezinio legenda
% hold off
end
```

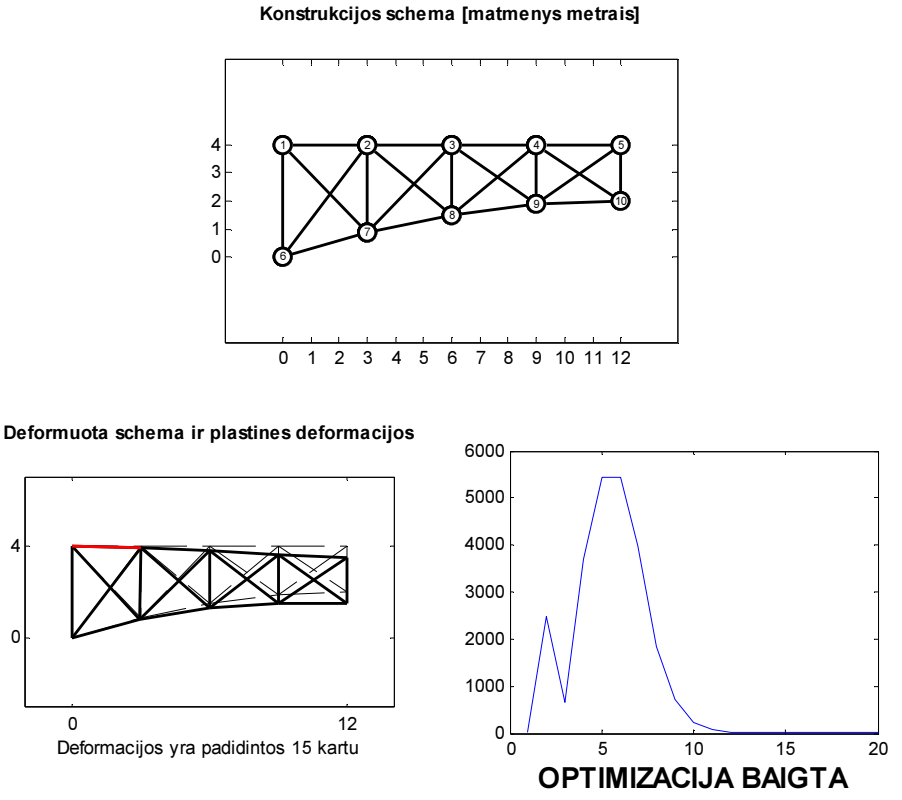
```
% cia aprasomi netiesiniai apribojimai
function [Cnq,Ceq] = san_apribojimai(x) %,n,k,sigmay,Vchi,Gama,bnq,Z)

global n k sigmay Gama Vchi bnq Z
% kadangi netiesiniu apribojimu nelygybemis nera, tai :
Cnq = [];
Ceq = [(x(1:2*n,1))'*(Z*x(1:2*n,1)-sigmay*[Gama; Gama].*[ones(n,k);
repmat(Vchi,1,k)]*x(2*n+1:2*n+k,1)-bnq(1:2*n,1));
(x(n+1:2*n,1).*((sigmay*Gama*x(2*n+1:2*n+k,1))-
(sigmay*Gama*x(2*n+1:2*n+k,1).*Vchi)))]; % eilute su nauja salyga
```

```
% si zemiau pateikta funkcija aprasys uzdavinio tikslo funkcija

function F = san_funkcija(x)
% san_funkcija(x,n,k,Gama,L)
% suminiai_ilgiai = sum(Gama.*repmat(L',1,k));

global n k  Gama L
F = sum(Gama.*repmat(L',1,k))*x(2*n+1:2*n+k);
```



A1 pav. Matlab programos sugeneruotas santvaros optimizavimo rezultatų vaizdas
Fig. A1. Figure of a truss optimization results generated by Matlab program