**Report\_8: Simulate the Deformation of a Rectangular Viscoelastic Object**

In this report, the deformation of a rectangular viscoelastic object in the 2D simulated. Equal and opposite horizontal force is being applied at **P8** and **P9** and **P1, P2, P3, P4** are fixed. Here the solver program and the result are included.

1. **ODE solver**

% Dynamic deformation of an elastic square object (4&times;4)  
% 正方形弾性物体の動的な変形 (4&times;4)  
% g, cm, sec  
  
addpath('../two\_dim\_fea');  
  
width = 30; height = 30; thickness = 1;  
m = 4; n = 4;  
[points, triangles] = rectangular\_object(m, n, width, height);  
  
% E = 1 MPa; c = 0.04 kPa s; rho = 1 g/cm^3  
Young = 10.0\*1e+6; c = 0.4\*1e+3; nu = 0.48; density = 1.00;  
[lambda, mu] = Lame\_constants(Young, nu);  
[lambda\_vis, mu\_vis] = Lame\_constants(c, nu);  
  
npoints = size(points,2);  
ntriangles = size(triangles,1);  
elastic = Body(npoints, points, ntriangles, triangles, thickness);  
elastic = elastic.mechanical\_parameters(density, lambda, mu);  
elastic = elastic.viscous\_parameters(lambda\_vis, mu\_vis);  
elastic = elastic.calculate\_stiffness\_matrix;  
elastic = elastic.calculate\_damping\_matrix;  
elastic = elastic.calculate\_inertia\_matrix;  
  
tp = 0.5; vpush = 0.8\*(height/3)/tp; hpush = 0.8\*(height/3)/tp; % add horizontal region  
th = 0.5;  
tf = 2.0;  
  
alpha = 1e+6;  
  
% pushing top region  
% 上部を押している  
A = elastic.constraint\_matrix([1,2,3,4,8,9]); %change the contraints that the force is  
b0 = zeros(2\*6,1);  
b1 = [ zeros(2\*4,1); -hpush; 0; hpush; 0]; % update the initial force in pushing region  
interval = [0, tp];  
qinit = zeros(4\*npoints,1);  
square\_object\_push = @(t,q) square\_object\_constraint\_param(t,q, elastic, A,b0,b1, alpha);  
[time\_push, q\_push] = ode15s(square\_object\_push, interval, qinit);  
  
% holding top region  
% 上部を保持している  
b0 = [ zeros(2\*4,1); -hpush\*tp; 0; hpush\*tp; 0 ]; % similarly update holding top region  
b1 = zeros(2\*6,1);  
interval = [tp, tp+th];  
qinit = q\_push(end,:);  
square\_object\_hold = @(t,q) square\_object\_constraint\_param(t,q, elastic, A,b0,b1, alpha);  
[time\_hold, q\_hold] = ode15s(square\_object\_hold, interval, qinit);  
  
% releasing top region  
% 上部を解放  
A = elastic.constraint\_matrix([1,2,3,4]);  
b0 = zeros(2\*4,1);  
b1 = zeros(2\*4,1);  
interval = [tp+th, tp+th+tf];  
qinit = q\_hold(end,:);  
square\_object\_free = @(t,q) square\_object\_constraint\_param(t,q, elastic, A,b0,b1, alpha);  
[time\_free, q\_free] = ode15s(square\_object\_free, interval, qinit);  
  
time = [time\_push; time\_hold; time\_free];  
q = [q\_push; q\_hold; q\_free];  
  
figure('position', [0, 0, 400, 400]);  
set(0,'defaultAxesFontSize',16);  
set(0,'defaultTextFontSize',16);  
  
clf;  
for t = 0:0.1:tp+th+tf  
 fprintf("time %f\n", t);  
 index = nearest\_index(time, t);  
 disps = reshape(q(index,1:npoints\*2), [2,npoints]);  
 elastic.draw(disps);  
 hold off;  
 xlim([-10,40]);  
 ylim([-10,40]);  
 xticks([-10:10:40]);  
 yticks([-10:10:40]);  
 pbaspect([1 1 1]);  
 grid on;  
 filename = strcat('4\_4/deform\_', num2str(floor(1000\*t),'%04d'), '.png');  
 saveas(gcf, filename, 'png');  
end  
  
clf('reset');  
ts = time(1);  
te = time(end);  
fr = 1;  
clear M;  
for t = 0:0.01:tp+th+tf  
 index = nearest\_index(time, t);  
 disps = reshape(q(index,1:npoints\*2), [2,npoints]);  
 elastic.draw(disps);  
 hold off;  
 xlim([-10,40]);  
 ylim([-10,40]);  
 xticks([-10:10:40]);  
 yticks([-10:10:40]);  
 pbaspect([1 1 1]);  
 title(['time ' num2str(t,"%3.2f")]);  
 grid on;  
 drawnow;  
 M(fr) = getframe(gcf);  
 fr = fr + 1;  
 disp(t);  
end  
M(fr) = getframe(gcf);  
  
v = VideoWriter('square\_object\_4\_4', 'MPEG-4');  
open(v);  
writeVideo(v, M);  
close(v);  
  
function dotq = square\_object\_constraint\_param(t,q, body, A,b0,b1, alpha)  
 %disp(t);  
  
 persistent npoints M B K;  
 if isempty(npoints)  
 npoints = body.numNodalPoints;  
 M = body.Inertia\_Matrix;  
 B = body.Damping\_Matrix;  
 K = body.Stiffness\_Matrix;  
 end  
  
 un = q(1:2\*npoints);  
 vn = q(2\*npoints+1:4\*npoints);  
  
 dotun = vn;  
  
 coef = [ M, -A; -A', zeros(size(A,2),size(A,2))];  
 vec = [ -K\*un-B\*vn; 2\*alpha\*(A'\*vn-b1)+(alpha^2)\*(A'\*un-(b0+b1\*t)) ];  
 sol = coef\vec;  
 dotvn = sol(1:2\*npoints);  
  
 dotq = [dotun; dotvn];  
end

1. **A picture containing text, crossword puzzle

   Description automatically generatedChart, line chart, scatter chart

   Description automatically generatedResult**

**Chart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generatedChart, radar chart

Description automatically generatedChart, radar chart

Description automatically generatedChart, radar chart

Description automatically generatedChart, radar chart

Description automatically generatedChart, radar chart

Description automatically generatedChart, radar chart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generatedChart, line chart

Description automatically generated**

**.**

**.**

**.**

**.**

**.**

**.**

**Chart, scatter chart

Description automatically generatedChart, line chart, scatter chart

Description automatically generated**

A picture containing text, crossword puzzle

Description automatically generated

[*Published with MATLAB® R2022b*](https://www.mathworks.com/products/matlab)