Supplementary material: low-resolution MATLAB topology optimization program with plate constraint filter

Yuji Wada, Tokimasa Shimada, Koji Nishiguchi, Shigenobu Okazawa, Makoto Tsubokura

1 Overview

The program `top_plate.m` performs low-resolution topology optimization using the filter described in the main text. The model is assumed to be a solid bar subjected to torsional loading, as in Example 1 of the main text. Place the program file in the working directory and execute the commands in accordance with the input arguments in the next section.

The topology optimization routine in this program has the following differences from those in the main text owing to implementation:

- First-order elements of the two-point Gauss–Legendre quadrature are used.
- The minimum density is set to 0.001 without using element deletion.
- MATLAB's sparse direct method solver is used.
- The long axis of the solid bar is in the z direction.
- Input force is set only at the edge points of the square bar.
- The number of iterations is 50.

The authors are not responsible for any failure of the optimization results except under the conditions given in the Example Results section.

1.1 Input argument

```
Usage:
top_plate(V0, filter_method, rh)
```

`V0` is the constraint of the volume fraction. Specify a range of 0.1–0.9. Because of resolution limitations, it will not work properly with low volume constraints such as those described in the main text.

`filter_method` is the filtering method switch. 0 is no filter, 1 is the explicit filter, 2 is the isotropic filter, and 3 is the plate constraint filter. When the explicit filter is used, the influence radius is always set to 1.5 times the mesh size.

`rh` is the filtering radius in millimeters. Since the resolution of the model is 6.25 mm and the length of the solid bar is 200 mm, use a filter radius of 5–100 mm. This value is not used when `filter_method` is 0 or 1.

In addition to this, the user can change Young's modulus, Poisson's ratio, dimensions in the x, y, and z directions, the number of iterations, and the resolution in lines 13–15 as needed, but the authors are not responsible for any problems resulting from these changes.

1.2 output files

The following files are output to the working directory:

`r***.vtk` is a finite element model and density distribution in the legacy VTK format. By opening with visualization software such as Paraview and adopting a threshold filter of a minimum value of 0.1, the intermediate or optimal shapes are visualized. Figure 1 shows the screenshot of the visualization in Paraview 5.9.0.

`obj.txt` is the history of the compliance in text format. `obj.png` is a simple plot of compliance vs iteration.

2 Example Results

The optimization and visualization are performed in MATLAB R2022a (Mathworks) and Paraview 5.9.0 (kitware), respectively. Using a workstation

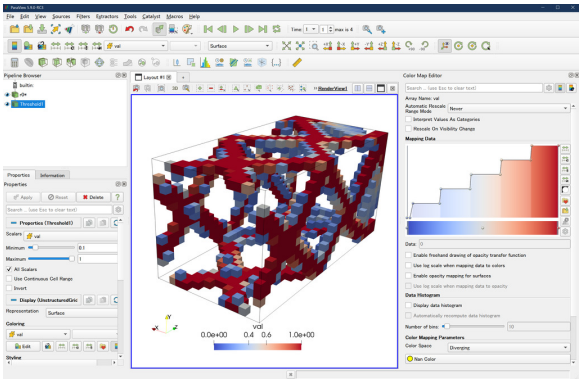

Fig. 1 Screenshot of the visualization.

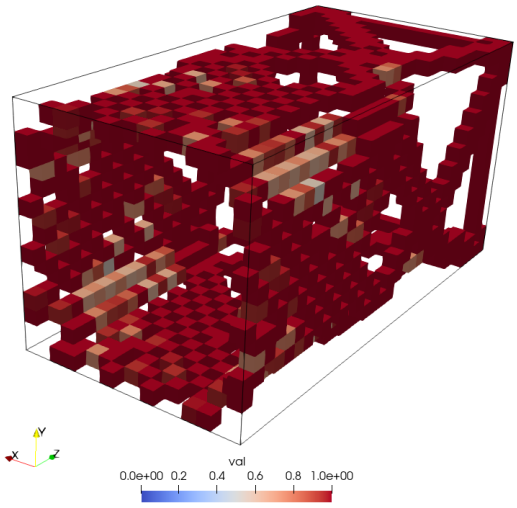

Fig. 2 Optimal shape without a filter.

with Intel Xeon W-2123 CPU (3.60 GHz, 4 core), the optimization time for 50 iterations is about 30 min, and the maximum memory used is 3 GB.

`top_plate(0.1, 0, 0)`
Figure 2 shows the optimal shape without a filter and with a volume constraint of 10%. The compliance is 54.2 nJ, so it is the stiffest example with a volume constraint of 10%, but the checkerboard distribution is observed.

`top_plate(0.1, 1, 0)`
Figure 3 shows the results of using the explicit

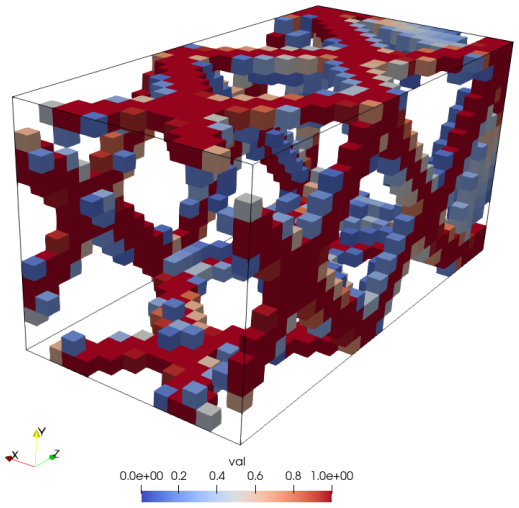

Fig. 3 Optimal shape with explicit filter.

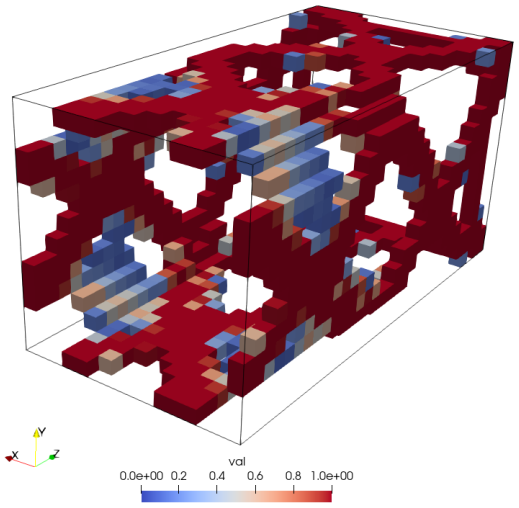

Fig. 4 Optimal shape with isotropic filter, r_h=30 mm.

filter and a volume constraint of 10%. The compliance is 61.4 nJ, 1.13 times higher than that without a filter. A frame structure with large members is obtained because of the small resolution.

`top_plate(0.1, 2, 30)`
Figure 4 shows the results of using the isotropic

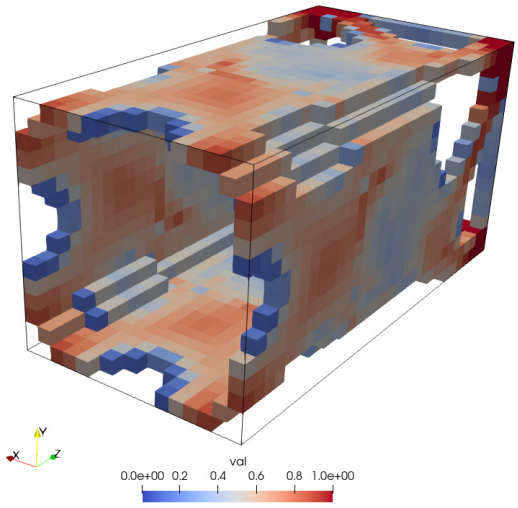

Fig. 5 Optimal shape with plate filter, r_h=100 mm.

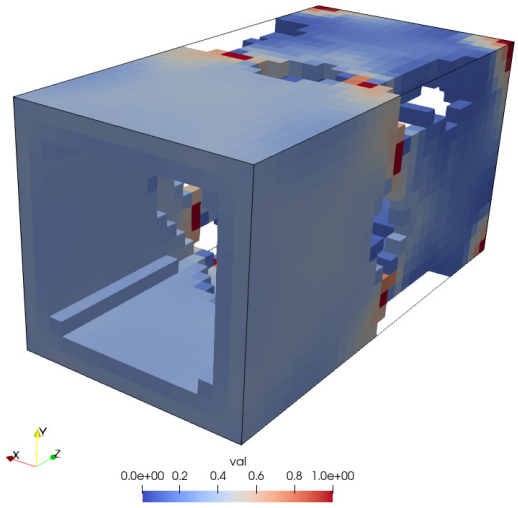

Fig. 6 Optimal shape with isotropic filter, r_h=100 mm.

filter with a radius of 30 mm and a volume constraint of 10%. The compliance is 61.7 nJ, 1.14 times higher than that without the filter. A simple frame structure is obtained.

`top_plate(0.1, 3, 100)`
Figure 5 shows the results of using the plate filter with a radius of 100 mm radius and a volume constraint of 10%. The compliance is 119 nJ, 2.20 times higher than that without the filter. A tubular structure with lower density is obtained.

`top_plate(0.1, 2, 100)`
Figure 6 shows the results of using an isotropic filter with a radius of 100 mm and a volume constraint of 10%. The compliance is 973 nJ, 18.0 times higher than that without the filter. There is a hole along the length axis, but most of the area is in an intermediate density state.

3 Program list of top_plate.m

```
function top_plate(V0,filter_method,rh)
%Topology optimization with plate constraint
%Example: top_plate(0.1,3,100)
%input:
%V0: volume fraction constraint: 0.1 -- 0.9
%filter_method: 0:no filter, 1:explicit(rh=1.5dx)
%               2:isotropic, 3:plate
%rh : filter radius (mm) : 5 -- 100
%output files:
%r***.vtk: density distribution, every 5 iter, VTK Legacy format
%obj.txt/.png : history of compliance

E=210e3; v=0.3;
lx=100; ly=100; lz=200;
n_iter=50; dx=100/16;

[nd,el,n,free,force] = meshgen(lx,ly,lz,dx);
[ke,cb,sn,dn,pvm]=get_ke3d(dx,E,v);
nel=n(4); ndof=n(6); nnode=n(5);
r=V0*ones(nel,1);
obj=zeros(n_iter,1);

for iter=1:n_iter
    K=sparse(ndof,ndof); u=zeros(ndof,1);
    for ie=1:nel
        map=[3*el(ie,:)-2 3*el(ie,:)-1 3*el(ie,:)];
        K(map, map)=K(map, map) + ke*r(ie)^3;
    end
    u(free)=K(free, free)\force(free);
    obj(iter)=0.5*u'*force;
    fprintf(1,'iter:%03d, obj:%e\n', iter, obj(iter));

    sens=zeros(nel,1); vm=zeros(nel,1); vv=zeros(3,3,nel);
    for ie=1:nel
        map=[3*el(ie,:)-2 3*el(ie,:)-1 3*el(ie,:)];
        ue=u(map);
        sens(ie)=-0.5*ue'*(3*r(ie)^2*ke)*ue;
        sig=cb*ue;
        vm(ie)=sqrt(sig'*pvm*sig);

        s=diag(sig(1:3));
        s(1,2)=sig(4); s(2,1)=sig(4);
        s(2,3)=sig(5); s(3,2)=sig(5);
        s(3,1)=sig(6); s(1,3)=sig(6);
        [v,d]=eig(s);
        [ds,ind]=sort(abs(diag(d)),'descend');
        vv(:,:,ie)=v(:,ind);
    end
    vm0=0.1*sum(r.*vm)/sum(r);
```

```
    h=0.5*(1+tanh(2*(vm-vm0)/vm0));

    if filter_method == 0
        senf = sens;
    elseif filter_method == 1
        senf = filter_explicit(sens,r,n);
    else
        A=sparse(nnode,nnode); ff=zeros(nnode,1);
        for ie=1:nel
            aa=zeros(8); fe = zeros(8,1);
            if filter_method == 2
                c=(rh^2/12)*eye(3);
            elseif filter_method == 3
                d=1.1*dx*eye(3) + rh*h(ie)*[1 0 0;0 1 0;0 0 0];
                c=vv(:,:,ie)*(d^2/12)*vv(:,:,ie)';
            end
            for ip=1:8
                dv=dx^3/8;
                aa = aa + dv*(dn(:,:,ip)'*c*dn(:,:,ip) + sn(:,ip)'*sn(:,ip));
                fe = fe - sens(ie)*sn(:,ip)*dv;
            end
            map=el(ie,:);
            A(map,map) = A(map,map) + aa;
            ff(map)= ff(map) + fe;
        end
        fu = A\ff;
        senf=zeros(nel,1);
        for ie=1:nel
            senf(ie) = mean(fu(el(ie,:)));
        end
        minsen=min(senf);
        senf = -(senf-minsen);
    end
    r=oc(r,senf,n,V0);
    if mod(iter,5)==0
        vtkout(sprintf('r%03d.vtk',iter),n,dx,r);
    end
end
plot(1:iter, obj(1:iter)); print -dpng obj.png
save -ascii obj.txt obj
end

function [nd,el,n,free,force] = meshgen(lx,ly,lz,dx)
n=zeros(6,1);
x=0:dx:lx; y=0:dx:ly; z=0:dx:lz;
n(1)=length(x); n(2)=length(y); n(3)=length(z);
n(4)=(n(1)-1)*(n(2)-1)*(n(3)-1); %element num
el=zeros(n(4),8);

[yy,xx,zz]=meshgrid(y,x,z);
n(5)=length(xx(:)); %node num
```

```
n(6)=3*n(5); %dof num
nd=zeros(n(5),3);
nd(:,1)=xx(:); nd(:,2)=yy(:); nd(:,3)=zz(:);

ie=0;
for zi=1:(n(3)-1)
    for yi=1:(n(2)-1)
        for xi=1:(n(1)-1)
            ie = ie + 1;
            el(ie,1)=xi+(yi-1)*n(1)+(zi-1)*n(1)*n(2);
            el(ie,2)=el(ie,1)+1;
            el(ie,3)=el(ie,2)+n(1);
            el(ie,4)=el(ie,1)+n(1);
            el(ie,5:8)=el(ie,1:4)+n(1)*n(2);
        end
    end
end

fix=[];
for yi=1:n(2)
    for xi=1:n(1)
        ni=xi+(yi-1)*n(1);
        fix=[fix,3*ni-2,3*ni-1,3*ni];
    end
end
free=setdiff(1:n(6),fix);

force=zeros(n(6),1);
n1= 1+( 1-1)*n(1)+(n(3)-1)*n(1)*n(2);
n2=n(1)+( 1-1)*n(1)+(n(3)-1)*n(1)*n(2);
n3=n(1)+(n(2)-1)*n(1)+(n(3)-1)*n(1)*n(2);
n4= 1+(n(2)-1)*n(1)+(n(3)-1)*n(1)*n(2);
force(3*n1-2)= 1;
force(3*n2-1)= 1;
force(3*n3-2)=-1;
force(3*n4-1)=-1;
end


function [ke,cb,sn,dn,pvm]=get_ke3d(dx,E,v)
lam=v*E/(1+v)/(1-2*v); mu= E/(2*(1+v));
c=zeros(6);  c(1:3,1:3)=lam+eye(3)*2*mu; c(4:6,4:6)=mu*eye(3);
pvm=zeros(6); pvm(1:3,1:3)=-0.5+1.5*eye(3); pvm(4:6,4:6)=3*eye(3);

xx=zeros(3,8);
xx(1,:)=[0 1 1 0 0 1 1 0];
xx(2,:)=[0 0 1 1 0 0 1 1];
xx(3,:)=[0 0 0 0 1 1 1 1];
xl=dx*xx; pm=2*xx-1; ipp=[pm/sqrt(3),[0;0;0]];

ke=zeros(24); cb=zeros(6,4);
```

```
sn=zeros(8,9); dn=zeros(3,8,9);
for i=1:9
    ip=ipp(:,i);
    N=zeros(1,8); dNds=zeros(3,8);
    N(:) = (1+pm(1,:)*ip(1))/2 .*(1+pm(2,:)*ip(2))/2 .* (1+pm(3,:)*ip(3))/2;
    dNds(1,:)= (pm(1,:))/2 .* (1+pm(2,:)*ip(2))/2 .* (1+pm(3,:)*ip(3))/2;
    dNds(2,:)= (1+pm(1,:)*ip(1))/2 .* (pm(2,:))/2 .* (1+pm(3,:)*ip(3))/2;
    dNds(3,:)= (1+pm(1,:)*ip(1))/2 .* (1+pm(2,:)*ip(2))/2 .* (pm(3,:))/2;
    dxds=dNds*xl';
    vol=det(dxds);
    dNdx = dxds\dNds;
    b=zeros(6,24);
    b(1,1:8)=dNdx(1,:); b(2,9:16)=dNdx(2,:); b(3,17:24)=dNdx(3,:);
    b(4,1:8)=dNdx(2,:); b(4,9:16)=dNdx(1,:);
    b(5, 9:16)=dNdx(3,:); b(5,17:24)=dNdx(2,:);
    b(6,17:24)=dNdx(1,:); b(6, 1:8 )=dNdx(3,:);

    sn(:,i)=N;
    dn(:,:,i)=dNdx;

    ke=ke +(i~=9)*vol*(b'*c*b);
    cb=(i==9)*c*b;
end
end

function senf = filter_explicit(sens,r,n)
senf=zeros(size(sens));
for zi=1:(n(3)-1)
for yi=1:(n(2)-1)
for xi=1:(n(1)-1)
    ie = xi + (yi-1)*(n(1)-1) + (zi-1)*(n(1)-1)*(n(2)-1);
    nume=0; deno=0;
    for zj=max(zi-1,1):min(zi+1,n(3)-1)
    for yj=max(yi-1,1):min(yi+1,n(2)-1)
    for xj=max(xi-1,1):min(xi+1,n(1)-1)

        je = xj + (yj-1)*(n(1)-1) + (zj-1)*(n(1)-1)*(n(2)-1);
        dist = sqrt((xj-xi)^2+(yj-yi)^2+(zj-zi)^2);

        nume = nume+sens(je)*r(je)*max(1.5-dist,0);
        deno = deno+r(je)*max(1.5-dist,0);
    end
    end
    end
    senf(ie) = nume/deno;
end
end
end
end

function rnew=oc(r,sens,n,V0)
```

```
senave=mean(abs(sens));
lamL = senave/1e3; lamH = senave*1e3; mlim = 0.2;
while ((lamH-lamL)/senave > 1e-4)
    lmid = sqrt(lamH*lamL);
    rnew = max(1e-3,max(r-mlim,min(1,min(r+mlim,r.*(-sens./lmid).^0.8))));
    if sum(rnew) - V0*n(4) > 0
        lamL = lmid;
    else
        lamH = lmid;
    end
end
end

function vtkout(fname,n,dx,val)
fid=fopen(fname,'w');
fprintf(fid, '# vtk DataFile Version 2.0\n');
fprintf(fid, 'opt-result\n');
fprintf(fid, 'ASCII\n');
fprintf(fid, 'DATASET STRUCTURED_POINTS\n');
fprintf(fid, 'DIMENSIONS %d %d %d\n',n(1),n(2),n(3));
fprintf(fid, 'ORIGIN %f %f %f\n',0,0,0);
fprintf(fid, 'SPACING %f %f %f\n',dx,dx,dx);

fprintf(fid, 'CELL_DATA %d\n',n(4));
fprintf(fid, 'SCALARS val float\n');
fprintf(fid, 'LOOKUP_TABLE default\n');
for i=1:n(4)
    fprintf(fid, '%e\n',val(i));
end
fclose(fid);
end
```