# Laser Driven Shock Waves

Authors: Pawan Veeresh & Raghav Sharma*

September 13, 2018

## NUMERICAL METHOD

We consider an axisymmetric two-dimensional disk to analyse a polymer sandwiched between two rigid glass plates. A laser beam heats a circular ring in the polymer. The only non-zero velocity is $v_r$ which is a function of the radial coordinate in an axisymmetric problem. The thickness of the polymer is small compared to other dimensions leading to a plane strain assumption in which,

$$\frac{\partial}{\partial z} \to 0 \Rightarrow u_z = 0 \Rightarrow \varepsilon_{zz} = 0$$

Also under an axisymmetric assumption,

$$\frac{\partial}{\partial \theta} \to 0 \Rightarrow u_\theta = 0 \Rightarrow \varepsilon_{\theta\theta} = \frac{u_r}{r}$$

which leads to,

$$[\varepsilon] = \begin{bmatrix} \varepsilon_{rr} \\ \varepsilon_{\theta\theta} \\ 0 \end{bmatrix}$$

The volumetric strain and deviatoric strains (assuming small strains) are,

$$\varepsilon_v = \mathrm{tr}(\varepsilon) = \varepsilon_{rr} + \varepsilon_{\theta\theta}$$

$$[\varepsilon'] = [\varepsilon] - \frac{\varepsilon_v}{3}[I]$$

The strain displacement matrix is,

$$B = \begin{bmatrix} \frac{\partial N}{\partial r} \\ \frac{N}{r} \\ 0 \end{bmatrix}$$

Using the principle of virtual work, the internal force can be calculated as,

$$f^{int} = 2\pi \int_0^R B^T[\sigma]\, r dr$$

*School of Matter, Transport and Energy, Arizona State University

where,

$$[\sigma] = [\sigma'] - p[I]$$

in which,

$$p = -K\frac{\ln(\varepsilon_v^e + 1)}{\varepsilon_v^e + 1} \quad \text{and} \quad [\sigma'] = 2G[\varepsilon']$$

$K$ and $G$ are the bulk and shear modulus respectively.

The laser heating is distributed by a Gaussian function along the radial direction resulting in a thermal strain described by,

$$\varepsilon_v^{tr} = Ae^{\left[-\frac{(r-r_o)^2}{w^2}\right]}$$

We discretised the momentum equation for a Lagrangian mesh. We solved the resulting non-linear problem with explicit time integration. We used the central difference method with a lumped mass matrix. We evaluated the velocities at every half-time step values. We found the accelerations and displacement at integer-time step values. Only the first time-step is different in that only a half-step is taken (using Forward Euler's method) to correctly account for the initial conditions on velocities.

In an explicit time integration, the time step must be below a critical value or the solution blows up due to a numerical instability. For 1D, 2-node elements that we considered for this analysis, the critical time step is,

$$\Delta t_{\text{critical}} \leq \frac{h}{c}$$

where $h$ is the uniform element size and $c$ is the wave speed given by

$$c = \sqrt{\frac{M}{\rho}}$$

in which wave modulus, $M$ is

$$M = K + \frac{4G}{3}$$

where $\rho$ is the density of the polymer.

Taking advantage of the Lagrangian mesh, the new value of density at every time step can be calculated by,

$$\rho_{current} = \frac{\rho_o}{1 + \varepsilon_v}$$

The variation in density as the pressure wave travels along the radial direction is given in fig. 1. We see a rise in density with the development of a shock front.

# CONVERGENCE STUDY

We compared the shape of the pressure wave when the inward wave had traveled half the distance to the centre of the plate for varying element sizes.

As the element size increases, the pressure wave spreads out across more elements. And, in doing so the peak pressure reduces. Pressure waves generated with smaller element sizes

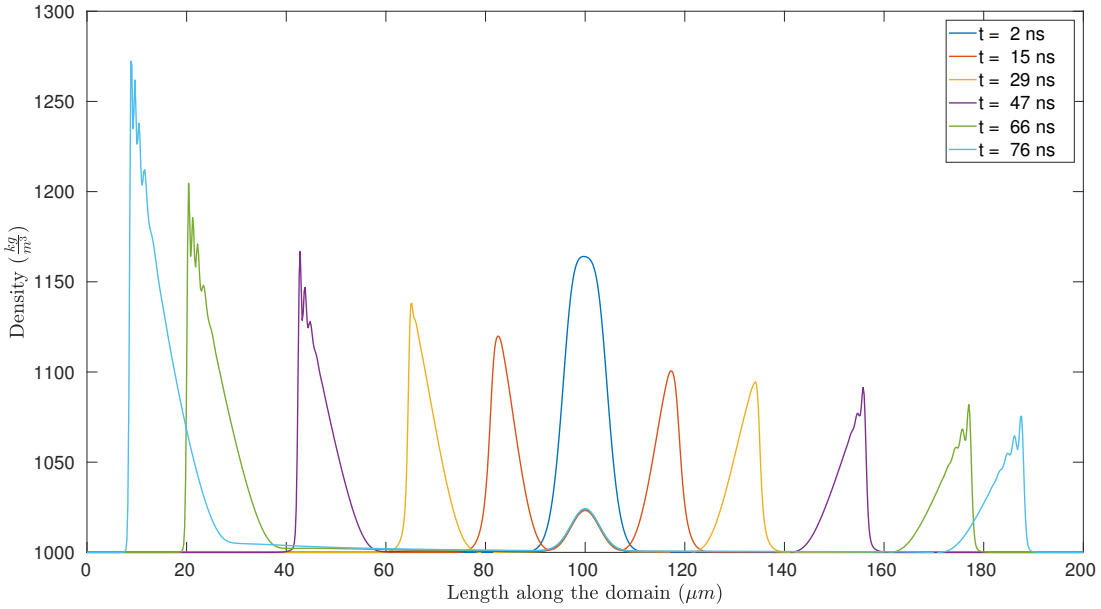


Figure 1: Density as a function of radial position.

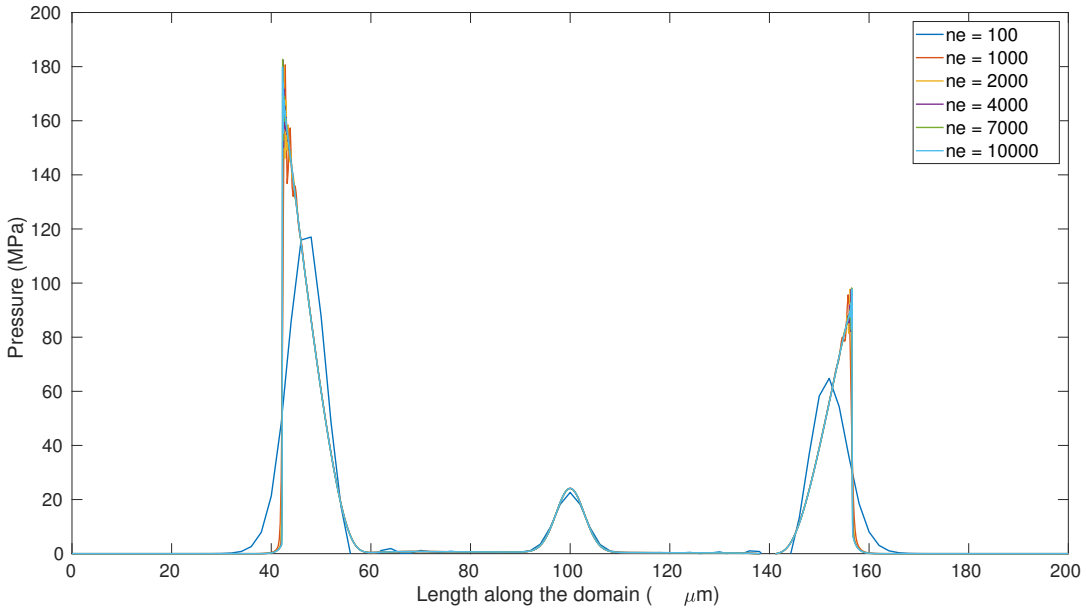


Figure 2: Pressure wave for different element sizes.

develop shock fronts more readily while being affected with greater oscillations in the elements containing high velocity gradients. Thus, smaller element sizes can be used with slightly larger values of artificial damping parameters as long as the dissipated energy is monitored (more on that in the following sections).

# ENERGY CONSERVATION

Instabilities are not easily discernible in non-linear problems unlike in linear problems where they grow exponentially and disrupt the solution. An energy balance check can be used to detect instabilities in non-linear computations. The relevant energies are computed as follows,

$$w^{int} = \frac{K}{2}\ln(1+\varepsilon_v)^2 + G[\varepsilon'].[\varepsilon']$$

$$w^{kin} = \frac{\rho}{2}\dot{u}_r^2$$

$$w^{dis} = \int Q\dot{\varepsilon_v}dt$$

and, the total energy is given by

$$w_{tot} = w^{int} + w^{kin} - w^{dis}$$

We evaluated all the energy densities at quadrature points and integrated using a one-point quadrature rule in each element. We integrated the dissipated energy over time as well. We made sure to calculate and compare energies using velocity at the same time step. We considered 100 elements of 2.0 $\mu$m length each. We plotted the energies at every 10 time steps (from a total of 250 steps with $\Delta t = 0.19$ ns)

As per fig. 3, the total energy in the system is conserved. All the energy at the first time step is internal energy. The dissipated energy increases monotonically. Any increase or decrease in internal energy manifests as kinetic energy of the moving particles. Thus, the energy in the system is conserved and the solution is stable.

# ARTIFICIAL DAMPING

Spurious oscillations arise due to discretisation errors in explicit finite element solution of wave propagation problems. We add artificial bulk viscosity to the pressure to smooth it out. The damping pressure is,

$$Q = \begin{cases} b_1\rho ch|\dot{\varepsilon_v}| + b_2\rho h^2\dot{\varepsilon_v}^2 & \text{for } \dot{\varepsilon_v} < 0 \\ 0 & \text{else} \end{cases}$$

where $\dot{\varepsilon_v}$ is the volumetric strain rate.

The damping pressure has a linear term and a quadratic term which have a linear and quadratic dependence on the volumetric strain rate. The linear term is introduced to damp ringing in the highest element frequency. The quadratic term spreads the shock front across

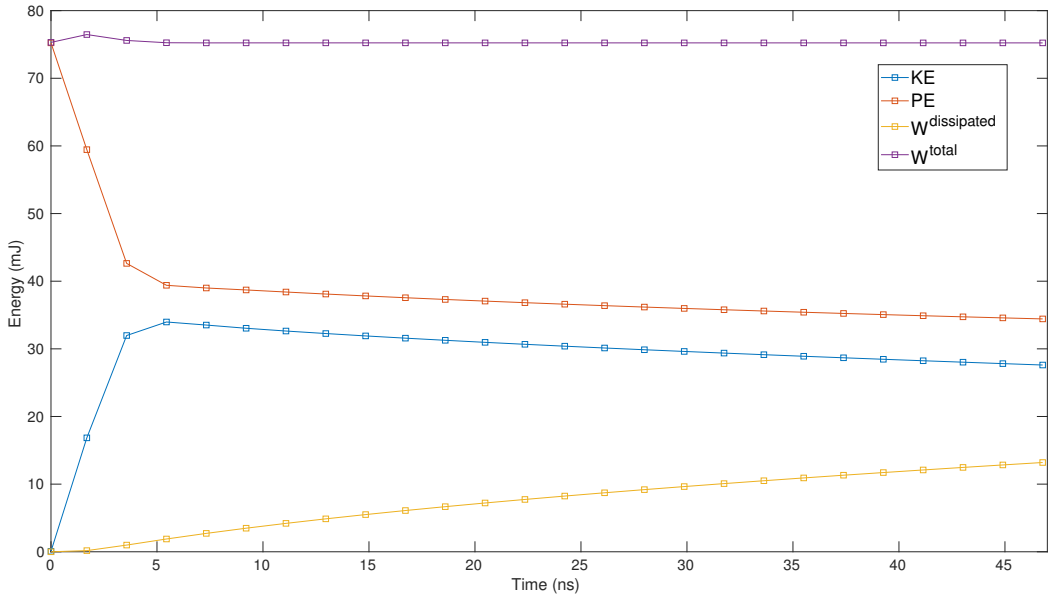


Figure 3: Conservation of system energy as a function of time.

several elements and is introduced to prevent elements from collapsing on themselves under extremely high velocity gradients.

The damping parameters used in the previous sections were the ABAQUS default values ($b_1 = 0.06$ and $b_2 = 1.2$). To check their validity, we compared the amount of energy dissipated to the amount of oscillation in the pressure wave for different values of $b_1$ and $b_2$ at a fixed instant in time (when the inward wave had traveled half the distance to the centre of the plate.).

We kept $b_2$ constant at 1.2 and varied $b_1$ for a domain containing elements of size 0.20 $\mu$m (total time = 47 ns and $\Delta$t = 0.02 ns).

Fig. 4 shows that with decreasing values of $b_1$, the peak pressure gets noisier i.e. increasing values of $b_1$ dampen the ringing effect in elements with very high velocity gradients. Fig. 5 points out the increasing amount of energy dissipated with increasing $b_1$ values.

Alternatively, we kept $b_1$ constant at 0.06 and varied $b_2$ for similar simulation parameters.

Fig. 6 shows that with increasing $b_2$ values, the shock front spreads out over more elements. Fig. 7 points out the increasing amount of energy dissipated with increasing $b_2$ values.

Figs. 4 - 7 indicate that the damping pressure is more sensitive to the changes in $b_1$ than to the changes in $b_2$. So, we kept the ratio of $b_1$ and $b_2$ constant for different values of $b_1$.

Shock waves are very steep but finite gradients in the solution. At the shock wave, the solution is intentionally distorted by adding bulk viscosity leading to lower than normal gradients. This smearing effect amplifies with increasing values of $b_2$, with $b_1 = 1.0$ and $b_2 = 20.0$ causing a significant reduction in peak pressure as shown in fig. 8. This combined with the result in fig. 9 that illustrates the increase in dissipated energy with increasing values of $b_1$ and $b_2$, tells us that values larger than $b_1 = 1.0$ and $b_2 = 20.0$ must be avoided. Therefore,

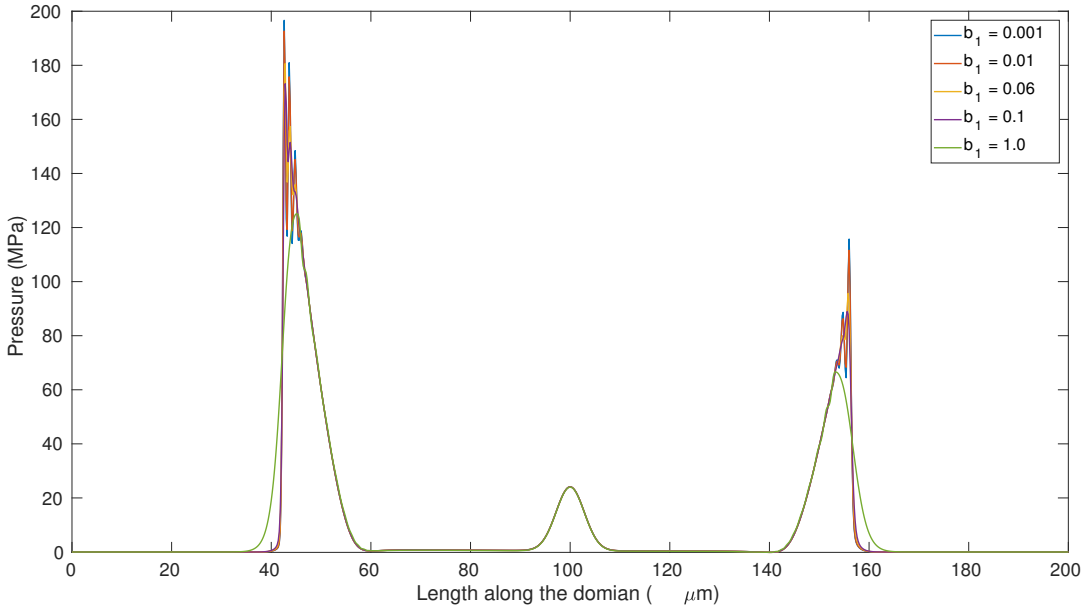


Figure 4: Pressure wave for different values of $b_1$.

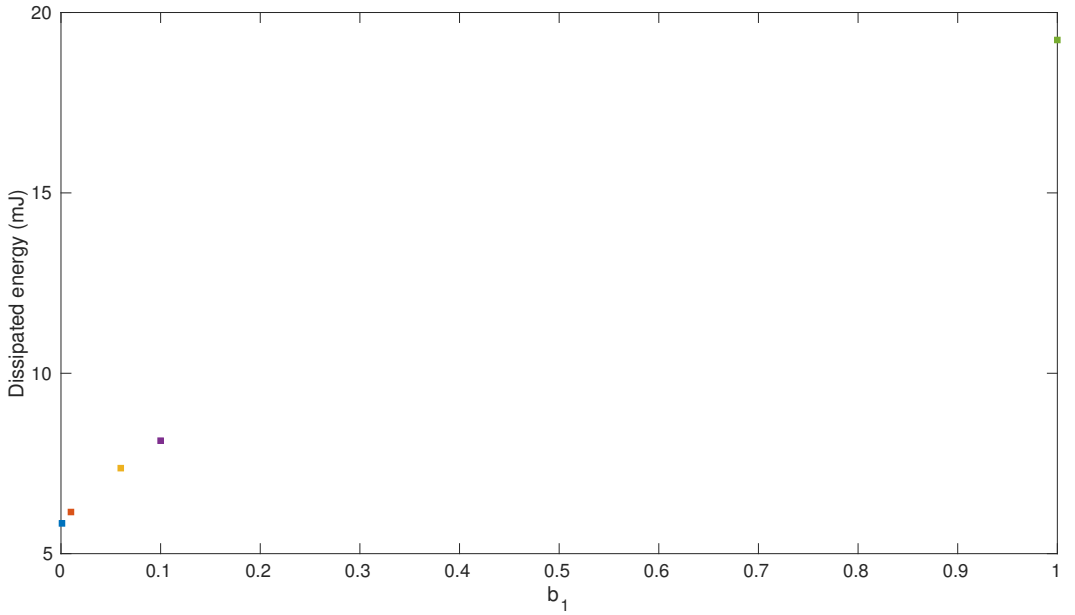


Figure 5: Dissipated energy for different values of $b_1$.

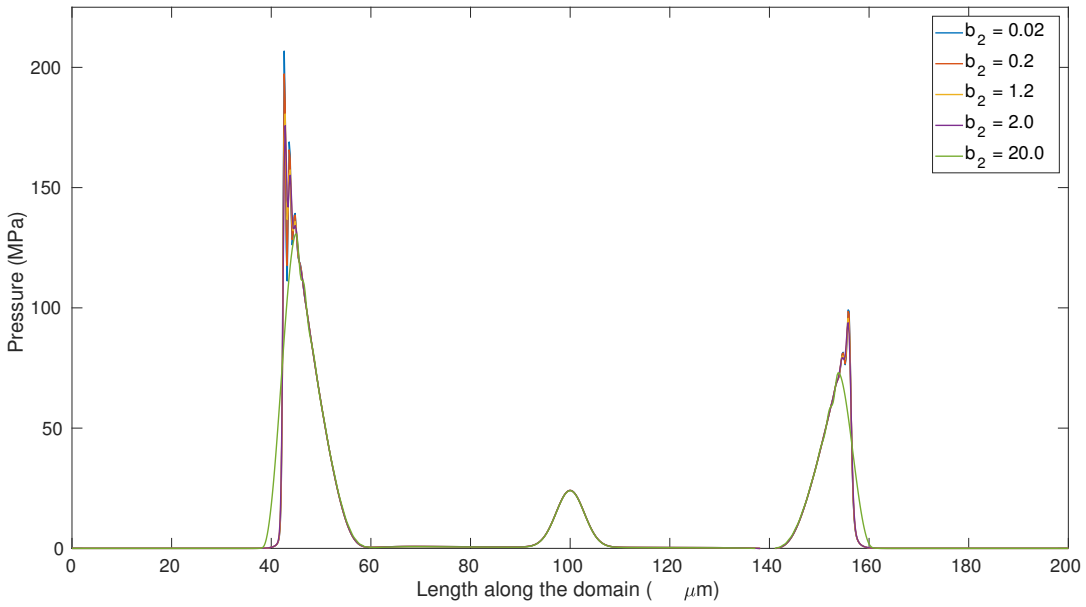


Figure 6: Pressure wave for different values of $b_2$.

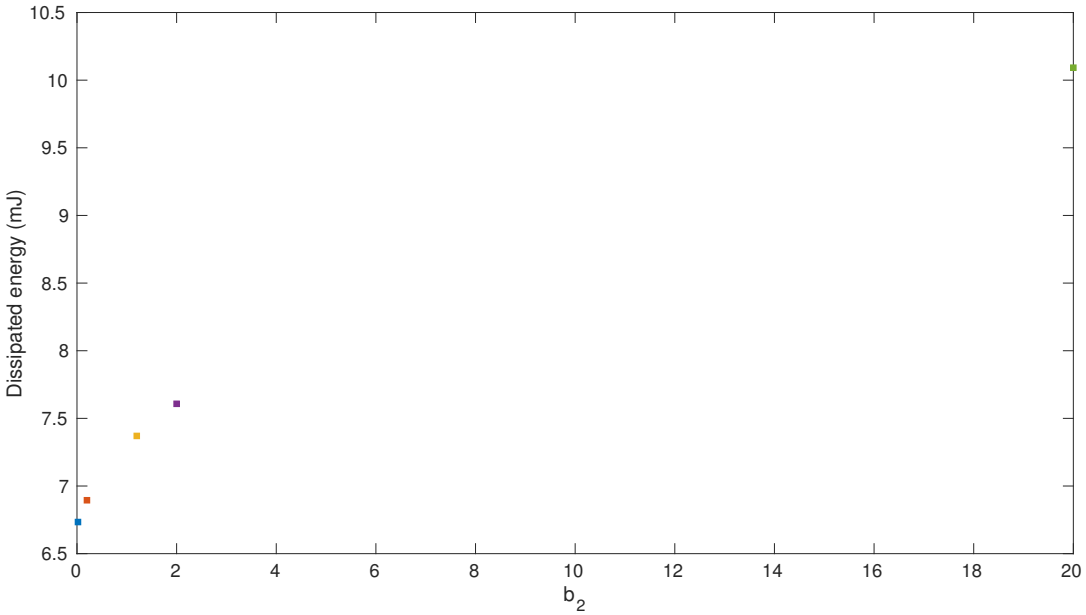


Figure 7: Dissipated energy for different values of $b_2$.

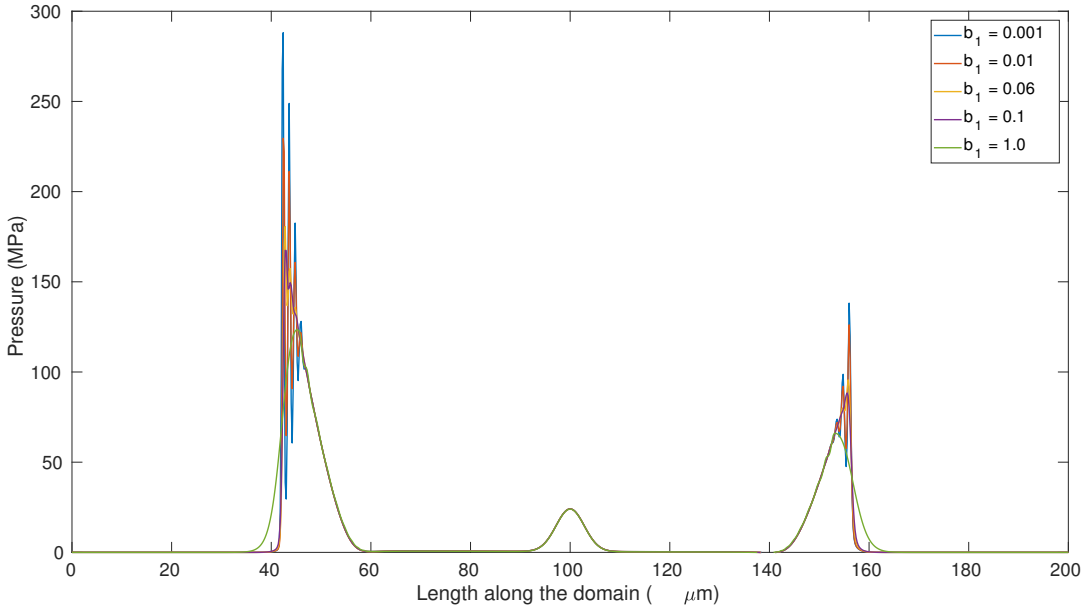


Figure 8: Pressure wave for constant ratio of $b_1$ and $b_2$.

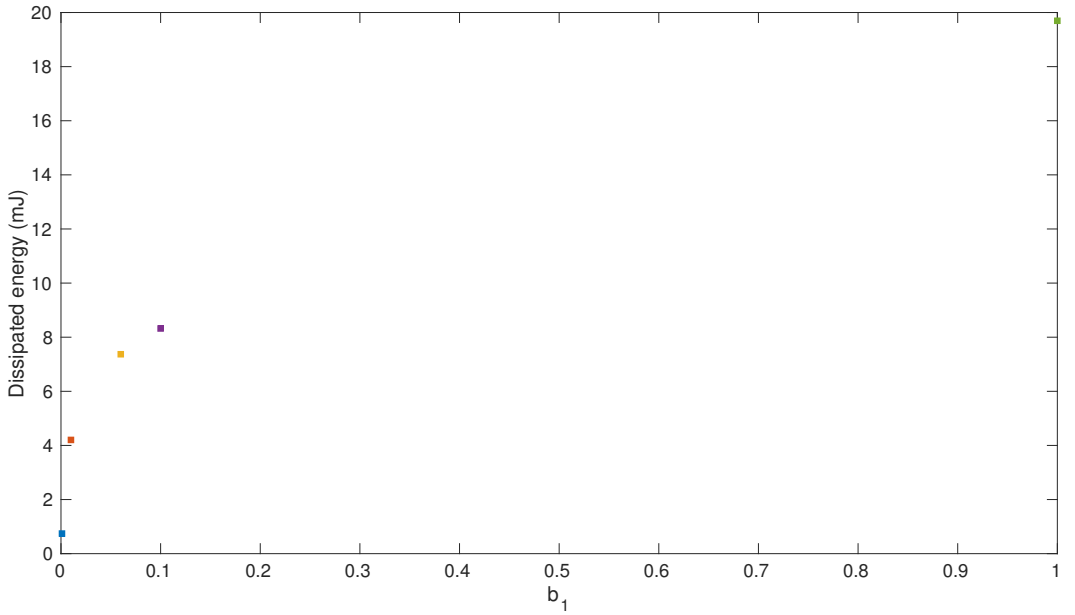


Figure 9: Dissipated energy for constant ratio of $b_1$ and $b_2$.

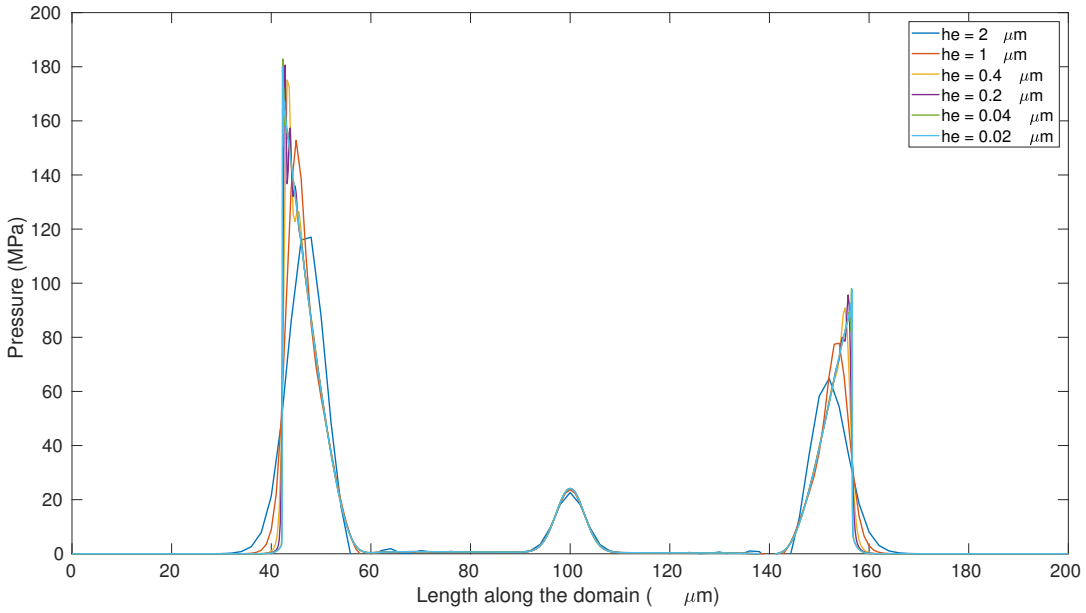


Figure 10: Pressure wave for different element sizes.

the ABAQUS default values of $b_1 = 0.06$ and $b_2 = 1.2$ are a good choice for coarser meshes ($\approx 100$ elements).

Ultimately, we checked the amount of energy that was dissipated for decreasing element size using the ABAQUS default values of $b_1 = 0.06$ and $b_2 = 1.2$.

Fig. 10 tells us that the amount of oscillations in the solution increase with reducing element size. And, fig. 11 indicates that the dissipated energy reduces slightly with reducing element size. The slope of dissipated energy vs. element size (represented on a log-log scale in fig. 11) curve is close zero for element sizes varying from 2 $\mu m$ to 0.02 $\mu m$. These results can be explained by the linear and quadratic dependence (in the linear and quadratic terms respectively) of damping pressure on element size and volumetric strain rate. A smaller element size leads to less dissipated energy but this decrease is offset by the increase in volumetric strain rate as the shock front develops.

# INITIAL LOADING

We changed the initial thermal strain value (A) and looked at its effect on the pressure distribution as the wave traveled. From fig. 12 we noticed that the inward wave grew in intensity (the top-half of the leading edge of the pressure wave catches up with the bottom-half) and developed a shock front while the outward wave dissipated. The inward wave collapsed on itself generating about 1 GPa pressure at the centre of the disk while the outward wave was negligible in comparison at the outer edge of the disk.

We chose a larger value for A ($= 0.8$) and noticed (fig. 13) that the shock fronts developed quicker and the inward wave collapsed with a much larger intensity ($> 2$ GPa) owing to a head-start in intensity from the initial loading.

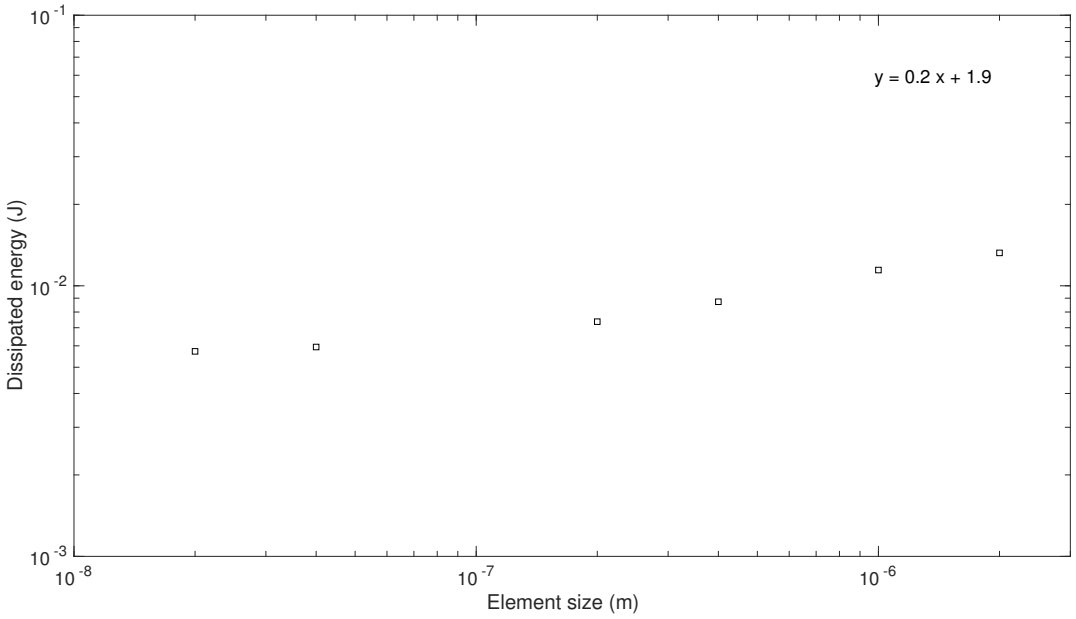


Figure 11: Dissipated energy for different element sizes.

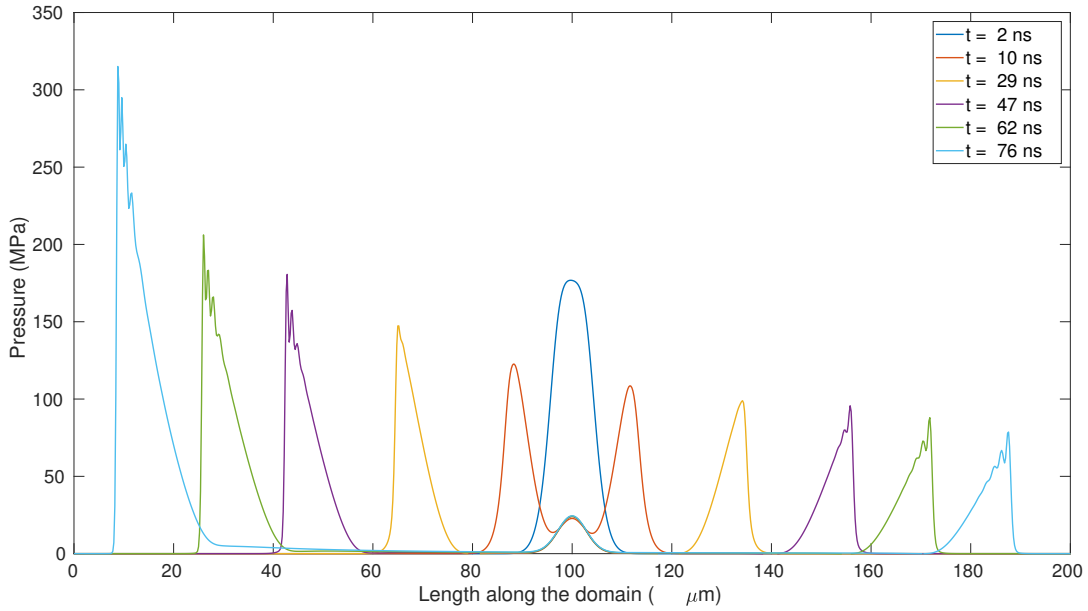


Figure 12: Pressure wave at different times for A = 0.2.

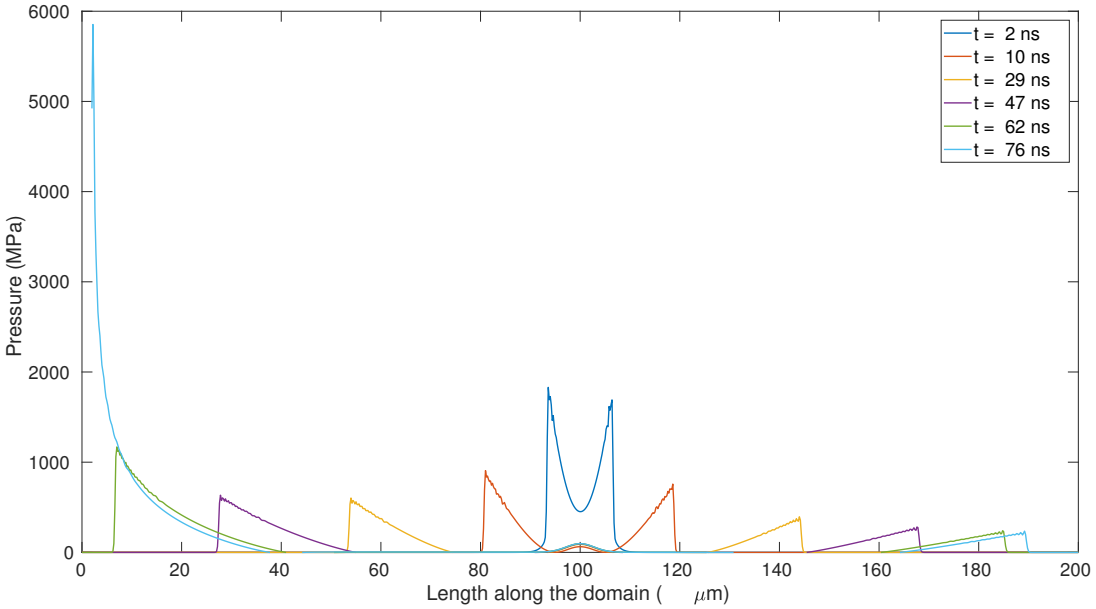


Figure 13: Pressure wave at different times for A = 0.8.

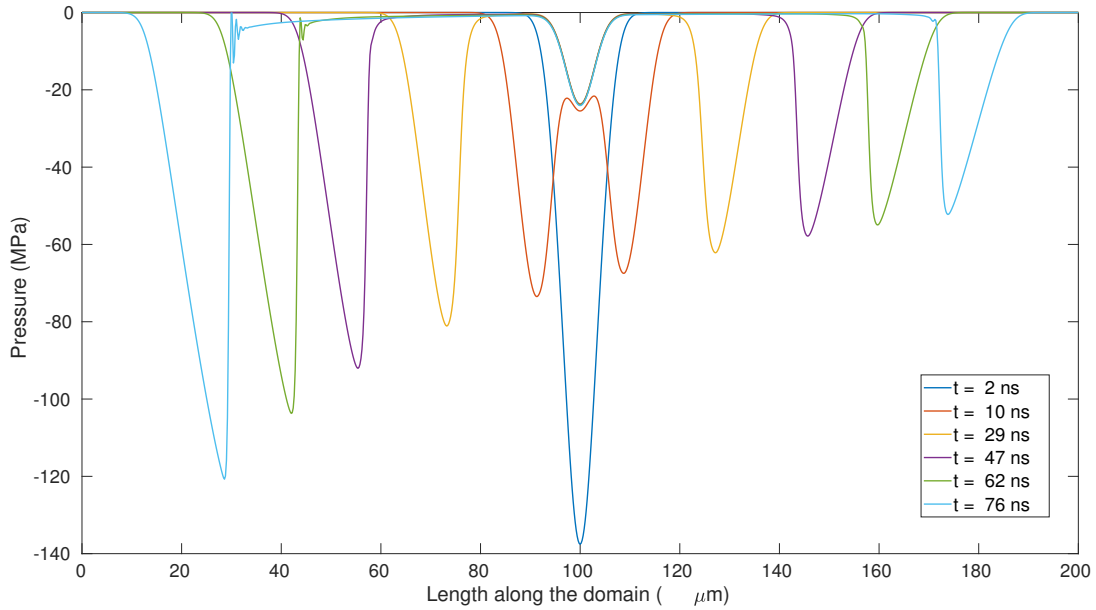


Figure 14: Pressure wave at different times for A = -0.2.

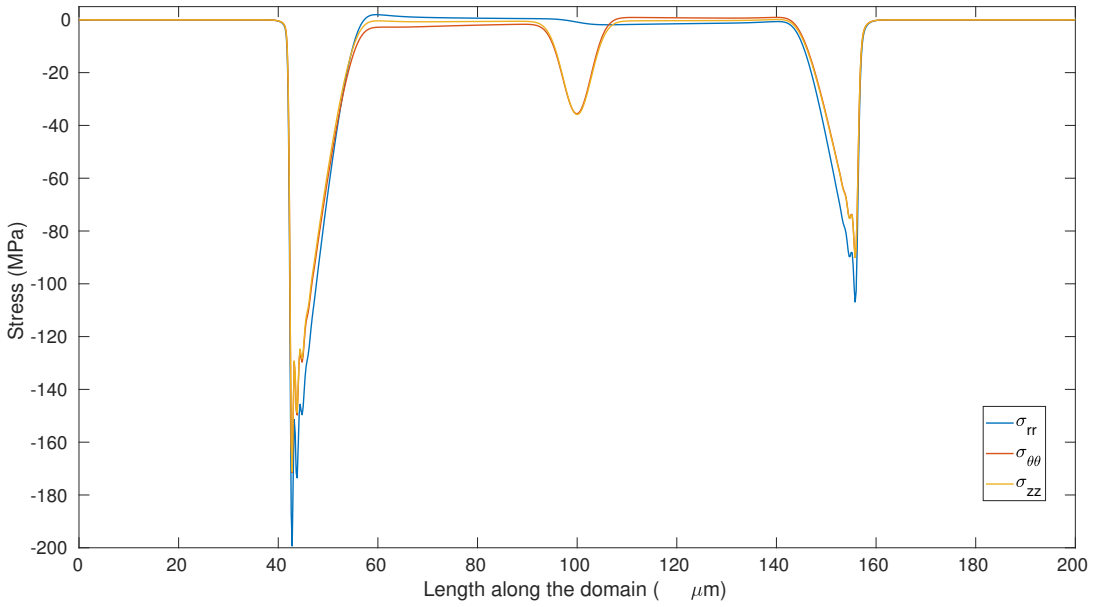


Figure 15: Stress components at t = 47 ns.

We then chose a negative value for the initial thermal load which generated tensile waves in the disk, fig. 14. Being in tension, these waves did not develop shock fronts but instead, the leading edge slowed down with time. This is in keeping with the fact that shock waves are developed only in compression.

# BONUS

In all the previous plots, we noticed a non-oscillatory stationary wave at the centre of the simulation domain. This wave neither grew in magnitude nor moved for the entire length of the simulation. To investigate this further, we looked into the nodal stresses $\sigma_{rr}$, $\sigma_{\theta\theta}$ and $\sigma_{zz}$.

From fig. 15 it can be seen that the non-oscillatory stationary wave manifests in the hoop stress and the stress along the disk thickness. With the displacements (particles) in those directions equal to zero, the applied thermal strain develops into a pressure that does not grow or move with time. Another interesting aspect to this was that the amplitude of this wave was directly related to the initial thermal strain value i.e. reducing A by a factor of 2 reduced the amplitude of the wave by a factor of 2.

```
function [data] = hw_1(nn, b1, b2, A, d)
   if nargin < 3
       nn = 101;
       b1 = 0.06;
       b2 = 1.2;
       A = 0.2;
       d = 0.25;
   end
   m = 0.1;
   data.R = 200e-6;    % m
   data.ro = 100e-6;   % m
   data.w = 4e-6;      % m
   data.A = A;
   data.nn = nn;
   data.ne = data.nn - 1;
   data.rho = 1000.0;  % kg/m^3
   data.K = 1e9;       % Pa (N/m^2)
   data.G = 100e6;     % Pa (N/m^2)
   data.b1 = b1;
   data.b2 = b2;
   data.c = sqrt((data.K+4*data.G/3)/data.rho);
   data.h = data.R/data.ne;

   data.mesh.X = linspace(0, data.R, data.nn);
   data.mesh.conn = [1:data.nn-1; 2:data.nn];

   data.v = zeros(data.nn,1);
   data.a = zeros(data.nn,1);
   data.d = zeros(data.nn,1);

   data.energy.dissipated = 0.0;

   M = mass_matrix(data);

   data.dt = m * data.h/data.c;
   total_time = (d*data.R)/data.c;
   num_steps = ceil(total_time/data.dt);

   data.history = [];
   for ts = 1:num_steps
       t = (ts-1)*data.dt;
       data.energy.pe = 0.0;
       [f, data] = internal_force(data);
       data.a = f./M;
       if ts > 1
           data.v = data.v + 0.5*data.dt*data.a;
       end
       data.a(1) = 0.0;
       data.v = data.v + 0.5*data.dt*data.a;
       data.d = data.d + data.dt*data.v;

       if any(isnan(data.v))
```

```
            fprintf('Simulation unstable at step %d.\n', ts);
            break;
        end

        data.pressure(1) = 2*data.pressure(1);
        data.pressure(end) = 2*data.pressure(end);
        data.rho_current(1) = 2*data.rho_current(1);
        data.rho_current(end) = 2*data.rho_current(end);

        if mod(ts, 10) == 0 || ts == 1
            if nargin > 0
                continue;
            end
            data.energy.ke = kinetic_energy(data, M);
            data.energy.total = data.energy.ke + data.energy.pe + ...
                                data.energy.dissipated;
            data.history(end+1, :) = [t, data.energy.ke,
 data.energy.pe, ...
                                 data.energy.dissipated,
 data.energy.total];

            fprintf('ts ke      pe      W_dis   W_total \n');
            fprintf('%d  %6.5f %6.5f %6.5f %6.5f \n', ...
                    ts, data.energy.ke, data.energy.pe,
 data.energy.dissipated, ...
                    data.energy.total);

            % Energy conservation
            figure(1); clf;
            plot(1e9*data.history(:,1), 1e3*data.history(:,
[2,3,4,5]),'s-');
            legend('KE', 'PE', 'W^{dissipated}', 'W^{total}');
            pause(0.01);

            % Density
            figure(2); clf;
            plot(1e6*data.mesh.X, data.rho_current);
            pause(0.1);

            % Pressure
            figure(3); clf;
            plot(1e6*data.mesh.X, 1e-6*data.pressure);
            pause(0.01);

            % Stress
            figure(4); clf;
            plot(1e6*data.mesh.X, 1e-6*data.stress.rr); hold on;
            plot(1e6*data.mesh.X, 1e-6*data.stress.tt); hold on;
            plot(1e6*data.mesh.X, 1e-6*data.stress.zz);
        end
    end
end

function [M] = mass_matrix(data)
```

```
    M = zeros(data.nn,1);
    for c = data.mesh.conn
        Xe = data.mesh.X(c);
        for q = quadrature_1D(2)
            [N, dNdp] = shape(q);
            J0 = Xe*dNdp;
            X = Xe*N;
            M(c) = M(c) + N*data.rho*2*pi*X*det(J0)*q(end);
        end
    end
    assert(abs(sum(M)-(pi*data.R^2*data.rho)) < 1e-5);
end

function [fint, data] = internal_force(data)
    fint = zeros(data.nn,1);
    data.pressure = zeros(data.nn,1);
    data.stress.rr = zeros(data.nn,1);
    data.stress.tt = zeros(data.nn,1);
    data.stress.zz = zeros(data.nn,1);
    data.rho_current = zeros(data.nn,1);
    for c = data.mesh.conn
        de = data.d(c);
        Xe = data.mesh.X(c);
        ve = data.v(c);
        for q = quadrature_1D(1)
            [N, dNdp] = shape(q);
            J0 = Xe*dNdp;
            dNdX = dNdp/J0;
            X = Xe*N;
            B = [dNdX'; N'/X; [0 0]];
            eps_thermal_vol = data.A*exp(-((X-data.ro)^2)/(data.w^2));
            eps = B*de - (eps_thermal_vol/3)*[1;1;1];
            eps_vol = sum(eps);
            D = B*ve;
            eps_vol_dot = sum(D);
            Q = 0.0;
            if eps_vol_dot < 0.0
                Q =
 data.b1*data.rho*data.c*data.h*abs(eps_vol_dot) ...
                  + data.b2*data.rho*data.h^2*eps_vol_dot^2;
            end
            p = -data.K*(log(eps_vol+1)/(eps_vol+1)) + Q;
            eps_dev = eps - (eps_vol/3)*[1;1;1];
            T_dev = 2*data.G*eps_dev;
            T = T_dev - p*[1;1;1];
            wt = 2*pi*X*det(J0)*q(end);
            fint(c) = fint(c) - B'*T*wt;
            data.energy.dissipated = data.energy.dissipated ...
                                     - Q*eps_vol_dot*data.dt*wt;
            data.energy.pe = data.energy.pe +
 (0.5*data.K*log(1+eps_vol)^2 ...
                           + data.G*(eps_dev'*eps_dev))*wt;
            rho_current = data.rho / (1+eps_vol);
            data.pressure(c) = data.pressure(c) + N*p;
```

```
            data.stress.rr(c) = data.stress.rr(c) + N*T(1);
            data.stress.tt(c) = data.stress.tt(c) + N*T(2);
            data.stress.zz(c) = data.stress.zz(c) + N*T(3);
            data.rho_current(c)  = data.rho_current(c) +
 N*rho_current;
        end
    end
end

function [ke] = kinetic_energy(data, M)
    ke = 0.5*dot(data.v, M.*data.v);
end

function [q] = quadrature_1D(n)
    if n == 1
        q = [0; 2];
    elseif n == 2
        q = [[-1 1]/sqrt(3.0); [1 1]];
    end
end

function [N, dNdp] = shape(q)
    N = 0.5*[1-q(1); 1+q(1)];
    dNdp = 0.5*[-1; 1];
end
```

*Published with MATLAB® R2017a*

```
function [] = convergence()
    for nn = [101, 1001, 2001, 4001, 7001, 10001]
        fprintf('Working on %d elements.\n', nn);
        data = hw_1(nn, 0.06, 1.2, 0.2, 0.25);
        plot(1e6*data.mesh.X, 1e-6*data.pressure); hold on;
    end
    legend('ne = 100', 'ne = 1000', 'ne = 2000', 'ne = 4000', 'ne =
 7000', 'ne = 10000');
    xlabel('Domain (\mum)');
    ylabel('Pressure (MPa)');
end
```

*Published with MATLAB® R2017a*

```
function [] = damping()
    % Varying b1
    for b1 = [0.001 0.01 0.06 0.1 1.0]
        fprintf('Working on b1 = %d.\n', b1);
        data = hw_1(1001, b1, 1.2, 0.2);
        figure(1); clf;
        plot(b1, 1e3*data.energy.dissipated, 's'); hold on;
        figure(2); clf;
        plot(1e6*data.mesh.X, 1e-6*data.pressure); hold on;
    end
    legend('b_1 = 0.001', 'b_1 = 0.01', 'b_1 = 0.06', 'b_1 =
 0.1', 'b_1 = 1.0');

    % Varying b2
    for b2 = [0.02 0.2 1.2 2.0 20.0]
        fprintf('Working on b2 = %d.\n', b2);
        data = hw_1(1001, 0.06, b2, 0.2);
        figure(3); clf;
        plot(b2, 1e3*data.energy.dissipated, 's'); hold on;
        figure(4); clf;
        plot(1e6*data.mesh.X, 1e-6*data.pressure); hold on;
    end
    legend('b_2 = 0.02', 'b_2 = 0.2', 'b_2 = 1.2', 'b_2 = 2.0', 'b_2 =
 20.0');

    % Constant ratio of b1 & b2
    for b = [0.001 0.01 0.06 0.1  1.0;
             0.02 0.2   1.2 2.0 20.0]
        fprintf('Working on b1 = %d.\n', b(1));
        data = hw_1(1001, b(1), b(2), 0.2);
        figure(5); clf;
        plot(b(1), 1e3*data.energy.dissipated, 's'); hold on;
        figure(6); clf;
        plot(1e6*data.mesh.X, 1e-6*data.pressure); hold on;
    end
    legend('b_1 = 0.001', 'b_1 = 0.01', 'b_1 = 0.06', 'b_1 =
 0.1', 'b_1 = 1.0');

    % Energy vs. element size
    dissipated_energy = [];
    for nn = [101 201 501 1001 5001 10001]
        fprintf('Working on %d elements.\n', nn-1);
        data = hw_1(nn, 0.06, 1.2, 0.2);
        dissipated_energy(end+1) = data.energy.dissipated;
        figure(7); clf;
        plot(data.R/nn-1, data.energy.dissipated, 's'); hold on;
        figure(8); clf;
        plot(1e6*data.mesh.X, 1e-6*data.pressure); hold on;
    end
    he = data.R./[100 200 500 1000 5000 10000];
    legend('he = 2 \mum', 'he = 1 \mum', 'he = 0.4 \mum', 'he = 0.2
 \mum', 'he = 0.04 \mum', 'he = 0.02 \mum');
```

```
    figure(9); clf;
    loglog(he,dissipated_energy,'ks');
    p = polyfit(log(he), log(dissipated_energy), 1);
    str = sprintf('y = %.1f x + %.1f',p(1),abs(p(2)));
    text(he(end-1), max(dissipated_energy), str);
end
```

*Published with MATLAB® R2017a*

```
function [] = initial_loading()
i = 1;
for d = [0.01 0.075 0.15 0.25 0.35 0.4]
    fprintf('Working on d = %f.\n', d);
    data  = hw_1(1001, 0.06, 1.2, 0.2, d);
    t = ceil((d*data.R*1e9)/data.c);
    total_time{i} = sprintf('t =  %d ns', t);
    figure(1); clf;
    plot(1e6*data.mesh.X, data.rho_current); hold on;
    i = i + 1;
end
legend(total_time);
end
```

*Published with MATLAB® R2017a*