Computational Solid Mechanics
Assignment on 1D Plasticity

Kiran Sagar Kollepara
Computational Mechanics

Assumed mechanical properties common to plots:

$$E = 200\ GPa$$
$$\nu = 0.3$$
$$\sigma_y = 200\ MPa$$
$$\sigma_\infty = 300\ MPa$$

1. Perfect Plasticity
   Cyclic Step loading : $[0\ 1.5\sigma_Y\ 0\ -1.5\sigma_Y\ 0\ 1.5\sigma_Y]$ with each step active for 5 seconds. Strain history is generated for above loading assuming linear elasticity.

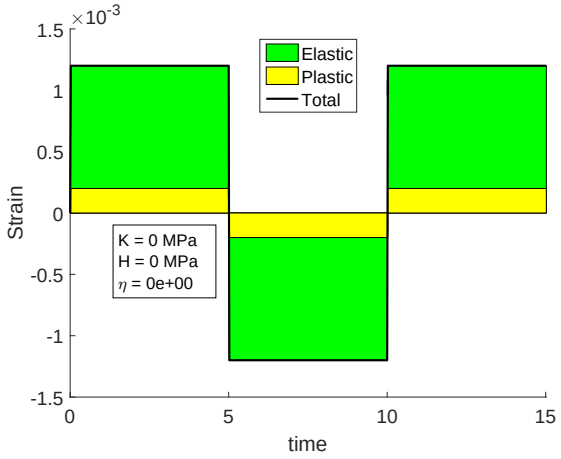

Figure 1: Strain history for perfect plasticity

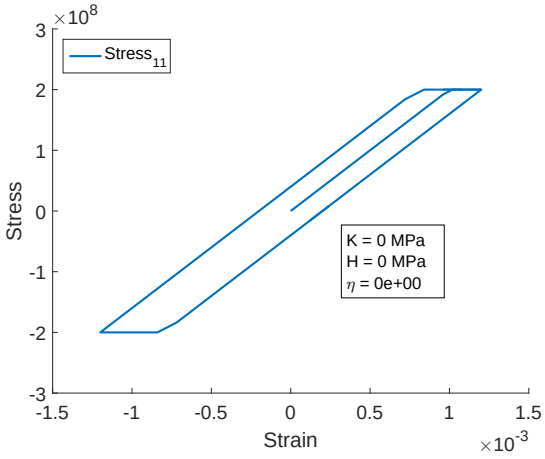

Figure 2: Stress vs. Strain for perfect plasticity

Observations:

- The stress in Figure 2 has zero slope as expected on the yield surface.
- The total stress σ has a non-zero slope because stress point moves along the yield surface, i.e. $\sigma = \sigma_Y\ \mathrm{sgn}(\sigma)$

For Rate Dependent model, with $\eta = 10^{11}$, the following behaviour is obtained:

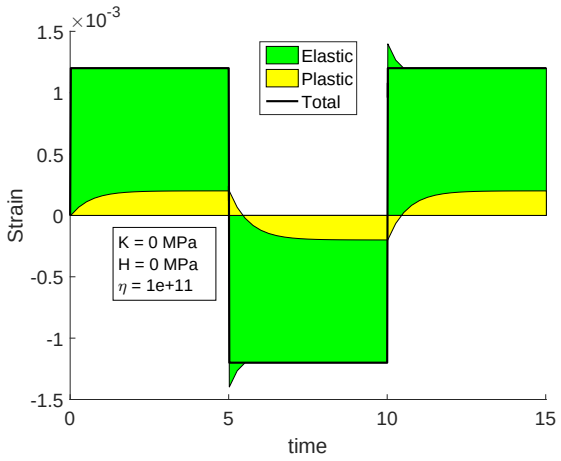

Figure 3: Strain history for Rate dependent perfect plasticity

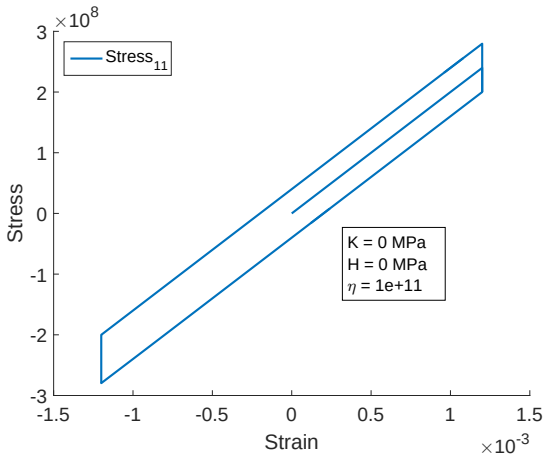

Figure 4: Stress vs. Strain for Rate dependent perfect plasticity

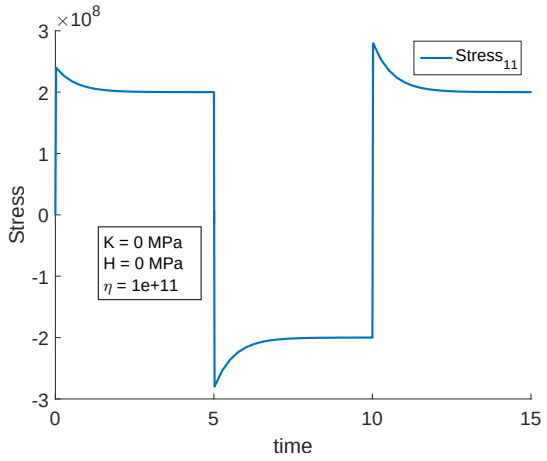

Figure 5: Stress vs. Time for Rate dependent perfect plasticity

Observations:

- Elastic and plastic strain curves have a non-zero settling time after each step, a clear indication of viscous effects.
- Stress vs. strain curve (Figure 4) show a near vertical behaviour during dwell time. This is corresponding to elastic strain reducing and plastic strain increasing during the dwell period. This can also be seen in Stress vs. time curve (Figure 5).

2. Linear Hardening
The same loading applied in this case as well.

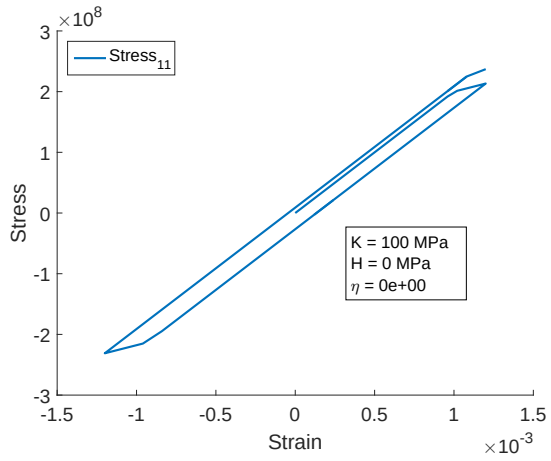

Figure 6: Stress vs. Strain for Linear Hardening

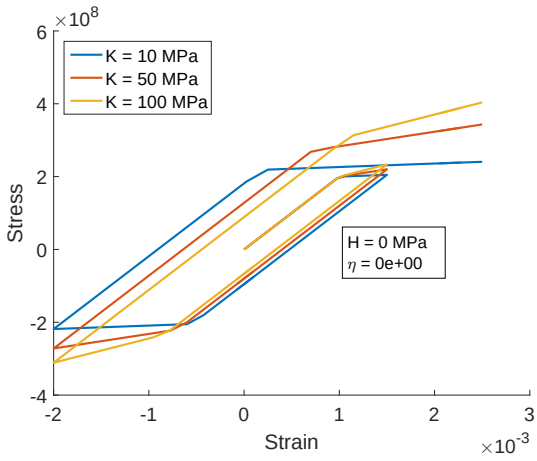

Figure 7: Stress vs. Strain for various K

Observations:

- The slope of stress $\frac{\sigma}{\sigma_Y}$ is of linear hardening modulus
- For various hardening modulii, it is observed that stress has higher values for higher modulus. (Figure 6)
- This gap between the curves of the two increases with each cycle, since each cycle has increases the size of yield surface. (Figure 7)

3. Exponential Hardening (Saturationa law)
Cyclic loading : $[0\ 5\sigma_Y\ 0\ -8\sigma_Y]$.

Here the coefficient δ of the exponential hardening law is calculated such that initial slope when hardening starts is same as that of Hardening modulus H. Figure 8 shows the response under such loading.

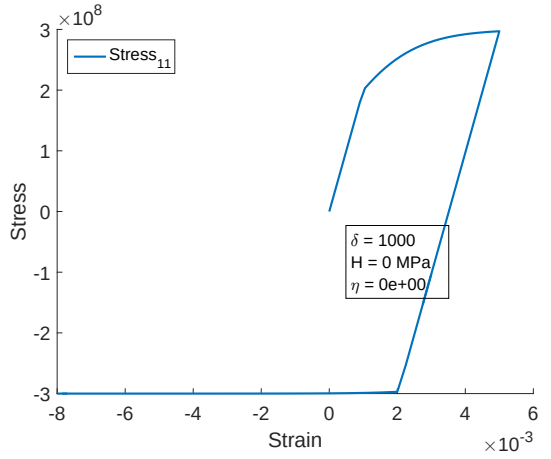

Figure 8: Stress vs. Strain for Non-linear Hardening with saturation law

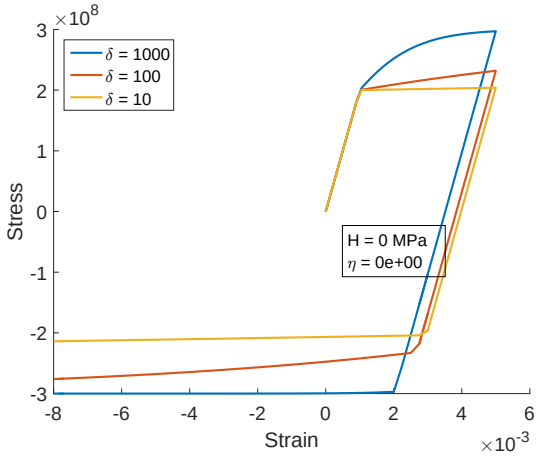

Figure 9: Stress vs. Strain for Non-linear Hardening with saturation law

Observations:

- On compression loading, the behaviour is similar to *perfect plasticity*. This is because of the yield surface is already saturated and cannot increase beyond this size.
- The above observation implies: closer σ_Y is to σ_∞, more is the behaviour like perfect plasticity.
- For higher values of δ, the yield surface saturates for a lower strain.

4. Kinematic Hardening
   Cyclic loading : $[0\ 3\sigma_Y\ 0\ -2\sigma_Y\ 0\ 2\sigma_Y]$.

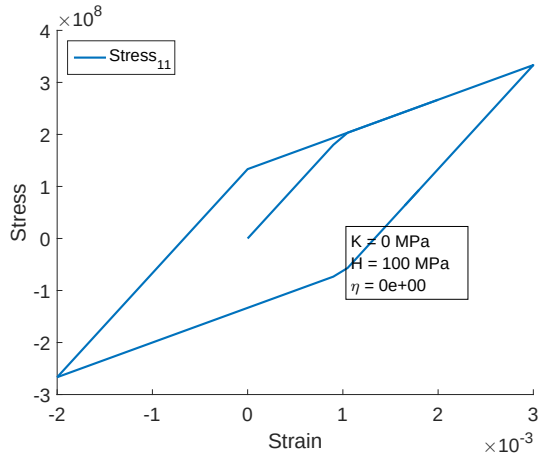

Figure 10: Stress vs. Strain for Kinematic Hardening

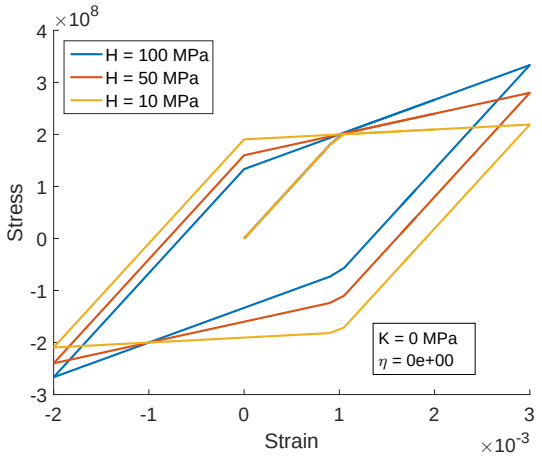

Figure 11: Stress vs. Strain for different Kinematic Hardening modulii

Observations:

- Loss of symmetry is observed clearly in Figure 10. Upon compressive loading, the stress vs. strain curve changes curve much earlier than the value of yield strength.

5. Linear Kinematic Hardening + Non-Linear Isotropic Hardening
   Cyclic loading in x-direction : $[0\ 3\sigma_Y\ 0\ -2\sigma_Y\ 0\ 2\sigma_Y]$.

   Observations:

- It is hard to identify the individual effects of non-linear hardening Hardening and Rate dependency.

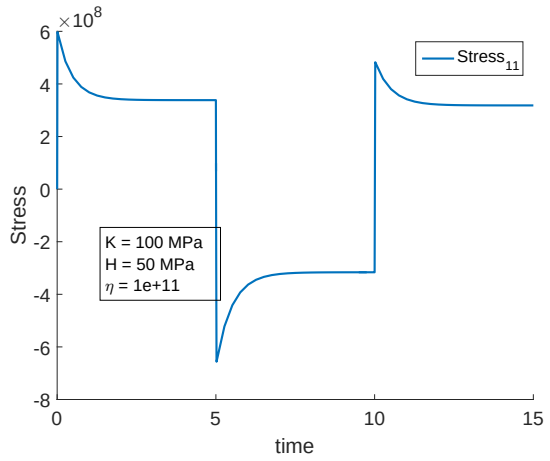

Figure 12: Stress vs. Time for Linear Kinematic + Non-Linear Isotropic Hardening + Viscosity

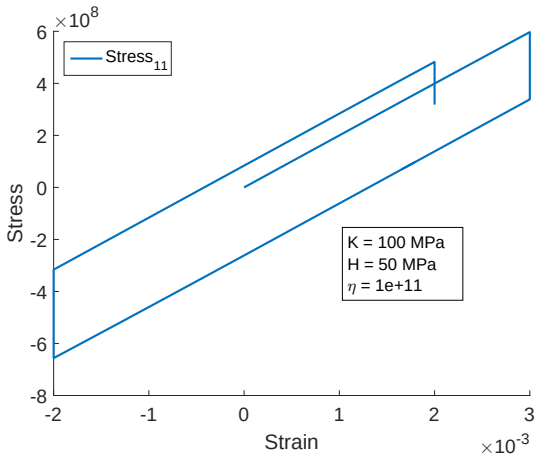

Figure 13: Stress vs. Strain for Linear Kinematic + Non-Linear Isotropic Hardening + Viscosity

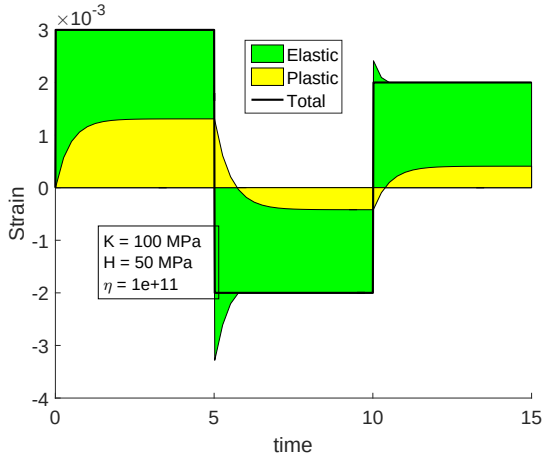

Figure 14: Strain vs. Time for Linear Kinematic + Non-Linear Isotropic Hardening + Viscosity

6. Recovery of Rate Independent from Rate Dependent Model

The below Figures 15 and 16 shows the recovery of rate independent model from rate dependent model. The figures are almost the same.

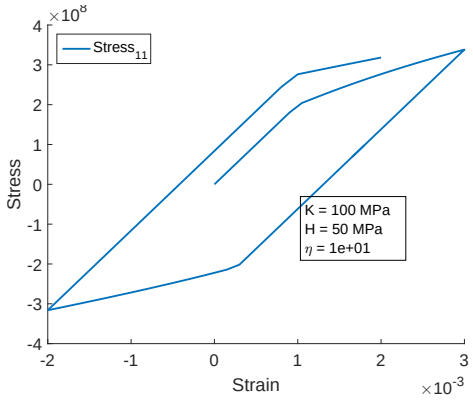

Figure 15: Stress vs. Time for Small value of Viscosity

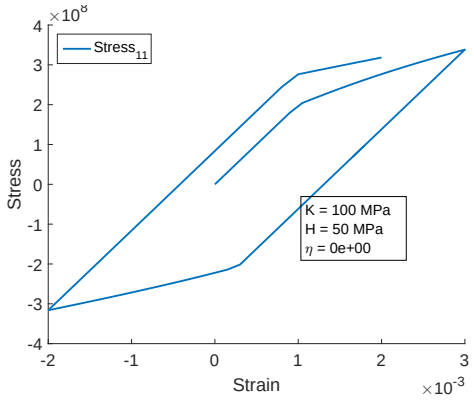

Figure 16: Stress vs. Strain for Rate Independent Model

Appendix
MATLAB codes on J2 Plasticity

1. Main m-file *main.m*

```
%% Material Properties
YOUNG           = 200e9 ;       NU              = 0.3 ;
ISO_HARD_MOD    = 100e9 ;       KIN_HARD_MOD    = 50e9 ;
YS              = 200e6 ;       YS_SAT          = 300e6 ;
HARD_TYPE       = 2 ; % 1 for linear, 2 for exponential
DELTA           = ISO_HARD_MOD/(YS_SAT- YS) ;
VISCOSITY       = 0 ;

Mat.young  = YOUNG ;                Mat.nu     = NU ;
Mat.isomod = ISO_HARD_MOD ;         Mat.kinmod = KIN_HARD_MOD ;
Mat.ys     = YS ;                   Mat.ys_sat = YS_SAT ;
Mat.htype  = HARD_TYPE ;            Mat.eta    = VISCOSITY ;

%% Load steps, time steps and Strain History
Load_step =  [0  3   3  0  -2   -2  2  2]*YS ;
steps_pls =   20*ones(1,length(Load_step)-1) ;
steps_time = [0 0.01 5 5.01 5.02 10 10.02 15] ;

% Generate strain history
Strain = [] ;
time   = [] ;
nsteps = sum(steps_pls)+1 ;
for i = 1:length(Load_step)-1
    Strain = Strain(1:end-1);
    time   = time(1:end-1);
    Strain = [Strain linspace(Load_step(i),Load_step(i+1),steps_pls(i)+1)] ;
    time   = [time linspace(steps_time(i),steps_time(i+1),steps_pls(i)+1)] ;
end
Strain = (Strain) / YOUNG ;

%% Construct storage variable
History = struct( 'sigma', cell(nsteps, 1)  'eps', num2cell(Strain(1:nsteps)'), ...
                  'eps_vp', cell(nsteps, 1),  'alpha', cell(nsteps, 1), ...
                  'q', cell(nsteps, 1), 'f', cell(nsteps, 1) ) ;
History(1).sigma = 0 ;      History(1).eps_vp = 0 ;
History(1).alpha = 0 ;      History(1).q      = 0 ;
History(1).f     = 0 ;
disp([num2str(nsteps),' steps in total'])

%% Time marching
for n = 1:nsteps-1
    del_t = time(n+1)-time(n) ;
    History(n+1) = eval_state_RD_1D(History(n),Strain(n+1),del_t,Mat) ;
end
```

2. Function to generate yield surface parameters *yield_surface_1D.m*

```
function  [ys,d_pi,dd_pi] = yield_surf_1D(Mat,alpha)
switch Mat.htype
    case 1
        ys = (Mat.ys + Mat.isomod*alpha) ;
        d_pi = Mat.isomod ;
        dd_pi = 0 ;
    case 2
```

```
        A = Mat.isomod/(Mat.ys_sat- Mat.ys) ;
        ys = Mat.ys_sat - (Mat.ys_sat- Mat.ys)*exp(-A*alpha) ;
        d_pi =  A*(Mat.ys_sat- Mat.ys)*exp(-A*alpha) ;
        dd_pi = -A^2*(Mat.ys_sat- Mat.ys)*exp(-A*alpha) ;
end
end
```

3. Function to generate data on next time step *eval_state_RD_1D.m*. Includes the return mapping algorithm.

```
function [new_state] = eval_state_RD_1D(state,eps_n1,del_t,Mat)

alpha           = state.alpha ;
q               = state.q ;
eps_vp          = state.eps_vp  ;

%Creating trial variable tr_state
tr_state.sigma  = Mat.young*(eps_n1 - eps_vp) ;
tr_state.eps    = eps_n1 ;
tr_state.eps_vp = state.eps_vp ;
tr_state.alpha  = alpha ;
tr_state.q      = q ;
tr_state.f      = abs(tr_state.sigma - q) - yield_surf_1D(Mat,alpha ) ;

if tr_state.f <= 0
    new_state = tr_state ;
else
    sigma      = tr_state.sigma ;
    iter       = 0 ;
    [~,d_pi,~] = yield_surf_1D(Mat,alpha);
    del_alpha   = tr_state.f/ (Mat.young + Mat.kinmod+ d_pi + Mat.eta/del_t) ;
    [~,d_pi,~] = yield_surf_1D(Mat,alpha+del_alpha);
    del_alpha   = tr_state.f/ (Mat.young + Mat.kinmod +d_pi + Mat.eta/del_t) ;

    del_alpha_old = 0;

    % Newton-Raphson iteration
    while abs((del_alpha_old - del_alpha)/del_alpha)>1e-2 || iter == 0
        [~,d_pi,dd_pi] = yield_surf_1D(Mat,alpha+del_alpha);
        g = tr_state.f-(Mat.young+d_pi+ Mat.kinmod+Mat.eta/del_t)*del_alpha ;
        gp = (Mat.young+d_pi+Mat.kinmod+Mat.eta/del_t) - dd_pi*del_alpha ;
        del_alpha_old = del_alpha ;
        del_alpha = del_alpha - g/gp ;
        iter        = iter+1 ;
    end

    alpha_n1    = alpha+del_alpha ;
    del_eps_vp  = del_alpha * sign(sigma - q) ;
    eps_vp_n1   = state.eps_vp + del_eps_vp ;
    sigma_old   = sigma ;
    sigma       = Mat.young*(eps_n1 - eps_vp_n1) ;
    q           = state.q + del_alpha*Mat.kinmod*sign(sigma - q) ;

    [new_ys] = yield_surf_1D(Mat,alpha_n1) ;

    % creating output structure
    new_state.sigma     = sigma ;           new_state.eps       = eps_n1 ;
    new_state.eps_vp    = eps_vp_n1 ;       new_state.alpha     = alpha_n1 ;
    new_state.q         = q ;               new_state.f         = abs(sigma - q) - new_ys ;
end
end
```