```matlab
function State = Mate25n (MatData,State)
%MATE25 hysteretic stress-strain relation after Menegotto-Pinto
%      with isotropic and kinematic hardening after Filippou et al. (EERC83-19)
% varargout = Mate25 (action,Mat_no,MatData,State)
%
% varargout : variable return argument list
%  varargout = MatData      for action 'chec'
%  varargout = State        for action 'init' with         fields sig, Et and Pres
%  varargout = State        for action 'stif' with updated fields sig, Et and Pres
%  varargout = State        for action 'forc' with updated field  sig     and Pres
%  varargout = [sig Post]   for action 'post'
%           where Et   = material tangent modulus
%                 sig  = current stress
%                 Pres = data structure with current values of material history variables
%                 Post = material post-processing information
% action   : switch with following possible values
%                'chec' material checks data for omissions
%                'data' material prints properties
%                'init' material initializes and reports history variables
%                'stif' material returns current stiffness and stress
%                'forc' material returns current stress only
%                'post' material stores information for post-processing
% Mat_no   : material number
% MatData  : data structure of material properties
% State    : current material state; data structure with updated fields eps, Past and Pres
%      .eps(:,1): total strains
%      .eps(:,2): strain increments from last convergence
%      .eps(:,3): strain increments from last iteration
%      .eps(:,4): strain rates
%      .Past    : history variables at last convergence
%      .Pres    : history variables at last iteration

% =========================================================================
% FEDEAS Lab - Release 2.3, March 2001
% Matlab Finite Elements for Design, Evaluation and Analysis of Structures
%
% Copyright (c) 1998, Professor Filip C. Filippou, filippou@ce.berkeley.edu
% Department of Civil and Environmental Engineering, UC Berkeley
% =========================================================================
% function contributed by Paolo Franchin & Alessio Lupoi, (c) January 2001

% Material Properties
% MatData.E   : initial modulus
%        .fy  : yield strength
%        .b   : strain hardening ratio
%        .R0  : exp transition elastic-plastic (default value 20)
%        .cR1 : coefficient for variation of R0 (default value 18.5)
%        .cR2 : coefficient for variation of R0 (default value 0.15)
%        .a1  : coefficient for isotropic hardening (default value 0)
%        .a2  : coefficient for isotropic hardening (default value 0)
%
% Material History Variables
% epmin : max strain in compression
% epmax : max strain in tension
% epex  : plastic excursion
% ep0   : strain at asymptotes intersection
% s0    : stress at asymptotes intersection
```

```
% epr   : strain at last inversion point
% sr    : stress at last inversion point
% kon   : index for loading/unloading

% extract material properties
b   = MatData.b;
R0  = MatData.R0;
cR1 = MatData.cR1;
cR2 = MatData.cR2;
a1  = MatData.a1;
a2  = MatData.a2;
E   = MatData.E;
fy  = MatData.fy;
% compute some material parameters
Es2 = b*E;       % hardening modulus
ey  = fy/E;      % yield strain

% material state determination
% ==========================================================================
% Retrieve history variables from Past
sig   = State.Past.sig;   % stress at last converged state
Et    = State.Past.Et;
epmin = State.Past.epmin;
epmax = State.Past.epmax;
epex  = State.Past.epex;
ep0   = State.Past.ep0;
s0    = State.Past.s0;
epr   = State.Past.epr;
sr    = State.Past.sr;
kon   = State.Past.kon;   % kon = 0 for virgin state, kon = 1 for loading,  kon = 2 for unloading
sigp  = sig;  % saved version of stress at last converged state

epm   = max(abs(epmin),abs(epmax));
epss  = State.eps(1,1); % total strain  (total strain at current iteration)
depss = State.eps(1,2); % total strain increment from last convergence

if kon==0 % the material is virgin
   if depss == 0
      sig  = 0;
      Et   = E;
   end
   if (depss>0)
      kon  = 1;
      ep0  = epm;
      s0   = fy;
      epex = epm;
      [sig,Et] = Bauschinger(epex,ep0,ey,R0,cR1,cR2,epss,epr,b,s0,sr);
   end
   if (depss<0)
      kon  = 2;
      ep0  = -epm;
      s0   = -fy;
      epex = -epm;
      [sig,Et] = Bauschinger(epex,ep0,ey,R0,cR1,cR2,epss,epr,b,s0,sr);
   end

else % material is damaged
```

```
    if (kon==1 & depss>0)|(kon==2 & depss<0) % keep loading in the previous step direction
       [sig,Et] = Bauschinger(epex,ep0,ey,R0,cR1,cR2,epss,epr,b,s0,sr);
    elseif (kon==1 & depss<0) % inversion from tensile to compressive
       kon  = 2;
       epr  = epss-depss;
       sr   = sigp;
       if epr>epmax  epmax = epr; end
       epm  = max(abs(epmin),abs(epmax));
       sst  = fy*a1*(epm/ey-a2);
       sst  = max(sst,0);
       ep0  = (sr+fy+sst-(E*epr+Es2*ey))/(Es2-E);
       s0   = Es2*(ep0+ey)-fy-sst;
       epex = epmin;
       [sig,Et] = Bauschinger(epex,ep0,ey,R0,cR1,cR2,epss,epr,b,s0,sr);
    elseif (kon==2 & depss>0) % inversion from compressive to tensile
       kon  = 1;
       epr  = epss-depss;
       sr   = sigp;
       if epr<epmin  epmin = epr; end
       epm  = max(abs(epmin),abs(epmax));
       sst  = fy*a1*(epm/ey-a2);
       sst  = max(sst,0);
       ep0  = (sr+Es2*ey-(E*epr+fy+sst))/(Es2-E);
       s0   = fy+sst+Es2*(ep0-ey);
       epex = epmax;
       [sig,Et] = Bauschinger(epex,ep0,ey,R0,cR1,cR2,epss,epr,b,s0,sr);
    end
end

% save history variables
State.Pres.sig   = sig;
State.Pres.Et    = Et;
State.Pres.epmin = epmin;
State.Pres.epmax = epmax;
State.Pres.epex  = epex;
State.Pres.ep0   = ep0;
State.Pres.s0    = s0;
State.Pres.epr   = epr;
State.Pres.sr    = sr;
State.Pres.kon   = kon;

State.sig = sig;
State.Et  = Et;

% ============================================================================

% ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function [sig,Et] = Bauschinger(epex,ep0,epy,R0,cR1,cR2,epss,epr,b,s0,sr)
% calculate stress and moduls

xi    = abs((epex-ep0)/epy);
R     = R0-(cR1*xi)/(cR2+xi); %
e_str = (epss-epr)/(ep0-epr); %
s_str = b*e_str+(1-b)*e_str/(1+abs(e_str)^R)^(1/R);
sig   = s_str*(s0-sr)+sr; %

dSdE  = b + (1-b) * (1-abs(e_str)^R/(1+abs(e_str)^R)) / (1+abs(e_str)^R)^(1/R);
```

```
Et    = dSdE*(s0-sr)/(ep0-epr);

% +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

```