```
function record = Static(P, MatData, MatState, algorithm_type , max_iter)
  record.P = P; % Saves the applied forces
  record.R = []; % Init internal force
  record.P_iter = []; % Record of R at every iteration
  record.U_iter = [];% Record of U at every iteration
  record.iter = []; % Record iterations per run
  record.MatData = MatData; % Saves the material data
  U_conv = 0; % Initialized last converged displacement as 0
  Delta_U = 0; % Initialized distance from last converged displacement was 0

   switch algorithm_type %  Checks which algorithm is used
      case "Newton" % N-R method
         tag = 1;
      case "ModifiedNewton" % Modified N-RMethod
         tag = 2;
      case "ModifiedNewton -initial" % Modified N-R Method with initial elastic tangent
         tag = 3;
   end

   for n = 1:numel(P)-1% Loop over load steps
      conv = 0; % Not converged at the beggining of the load
      j = 1; % Iteration counter reset to 1

      switch tag % Checks for modified N-R Method
         case 2
            algorithm_type = "ModifiedNewton";
            Ktan = MatData.A*MatState.Pres.Et/MatData.L;  % Tangent stiffness
         case 3
            algorithm_type = "ModifiedNewton -initial";
            Ktan = MatData.A*MatData.E/MatData.L;  % Initial stiffness
      end

      while ( j <= max_iter && conv == 0) % Loop over Newton-Raphson iterations while not converged
         MatState = Mate25n(MatData,MatState); % Update the material state
         R = MatData.A*MatState.Pres.sig; % Calculate teh internal resisting force
         Unb = P(n+1)-R; % Calculate the unbalance force
         record.Unb(n,j) = abs(Unb); % Record the unbalance force

         if (j > 1)
            record.U_iter = [record.U_iter,U_conv]; % Record the unbalance force path
            record.P_iter = [record.P_iter,R];% Record the unbalance force path
         end
        % Converged branch
         if (abs(Unb) < 1.e-5) % norm convergence criteria
            record.U(n+1) = U_conv; % Record the converged displacement
            record.R(n+1) = R;% Record the internal resisting force
            Delta_U = 0; % Reset total dispalcement from last displacement
            MatState.eps(1,2) = 0; % Reset total dispalcement from last displacement
            MatState.eps(1,3) = 0; % Reset incremental dispalcement from last displacement
            MatState.Past = MatState.Pres; % Commitees the present to the past. Total Strain remains
            conv = 1; % Convergence is true
            record.iter(n) =  j; % Record the iterations
         else
            if algorithm_type == "Newton" % Checks for Newton Algorithm to be used
               Ktan = MatData.A*MatState.Pres.Et/MatData.L;  % Tangent stiffness
            end
            if j == max_iter
```

```
58                 disp("Could not converged using " + algorithm_type + newline + "Switching to Newton-Raphson Method");
59                 j = 1; algorithm_type = "Newton"; % Switch to Newton if not converged
60             end
61             delta_U = Unb/Ktan; % Calculate the horizontal movement
62             Delta_U = Delta_U + delta_U; % Calculate total dispalcement from last displacement
63             U_conv = U_conv + delta_U; % Calculate converged dispalcement
64             MatState.eps(1,1) = U_conv/MatData.L;  % Total strain
65             MatState.eps(1,2) = Delta_U/MatData.L;  % Total incremental strain from last converged state
66             MatState.eps(1,3) = delta_U/MatData.L;  % Last incremental strain
67             j = j + 1; % Iteration counter
68             record.U_iter = [record.U_iter,U_conv]; % Record the unbalance force path
69             record.P_iter = [record.P_iter,P(n+1)];% Record the unbalance force path
70         end
71     end
72   end
73 end
```