```
function [y,t] = ode_solver(ode,initial,t0,t_end,N,method)
% initial muss ein Zeilenvektor sein!
dim = length(initial);
delta = (t_end-t0)/N ;
t = t0:delta:t_end;
y = [initial(:), zeros(dim,N-1)];

switch method
    case {1} % Euler-Cauchy
        for i=2:N+1
            y(:,i) = y(:,i-1) + delta*ode(t(i-1),y(:,i-1)) ;
        end
    case {2} % Verbesserter Euler-Cauchy
        for i=2:N+1
            y_step = y(:,i-1) + delta/2*ode(t(i-1),y(:,i-1));
            y(:,i) = y(:,i-1) + delta*ode(t(i-1)+delta/2,y_step) ;
            %disp('Verb. Euler')
        end
    case {3} % Prädiktor Korrektor
        for i=2:N+1
            y(:,i) = y(:,i-1) + delta*ode(t(i-1),y(:,i-1));
            for j=1:10
                y(:,i) = y(:,i-1) + delta/2*( ode(t(i-1),y(:,i-1)) +
 ode(t(i),y(:,i)) ) ;
            end
            %disp('Prädiktor')
        end
    case {4} % Höherer Prädiktor
        for i=2:N+1
        end
    case {5} % Runge Kutta 2.Ordnung (identisch zu verb. Euler-Cauchy)
        for i=2:N+1
            k1 = ode(t(i-1), y(:,i-1)) ;
            k2 = ode(t(i-1)+delta/2, y(:,i-1) + delta/2*k1) ;
            y(:,i) = y(:,i-1) + delta*k2 ;
        end
    case {6} % Runge Kutta 4.Ordnung
        for i=2:N+1
            k1 = ode(t(i-1), y(:,i-1)) ;
            k2 = ode(t(i-1)+delta/2, y(:,i-1) +delta/2*k1) ;
            k3 = ode(t(i-1)+delta/2, y(:,i-1) +delta/2*k2) ;
            k4 = ode(t(i), y(:,i-1) +delta  *k3) ;
            y(:,i) = y(:,i-1) +delta/6*(k1+2*k2+2*k3+k4) ;
        end
end
```

*Published with MATLAB® R2016a*