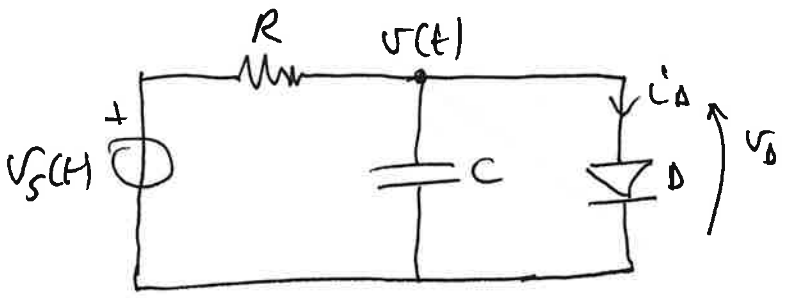
CIRCUITO NONLINEARE



$$i_D = I_0\left(e^{\frac{V_D}{V_T}}-1\right)$$

METODO DEI NODI PURO :

$$\frac{v(t)-V_S(t)}{R} + C\frac{dv(t)}{dt} + I_0\left(e^{\frac{v(t)}{V_T}}-1\right) = 0$$

$$\begin{cases} \dfrac{dv(t)}{dt} = -\dfrac{v(t)}{RC} - \dfrac{I_0}{C}\left(e^{\frac{v(t)}{V_T}}-1\right) + \dfrac{V_S(t)}{RC} \\ v(0) = V_0 \end{cases}$$

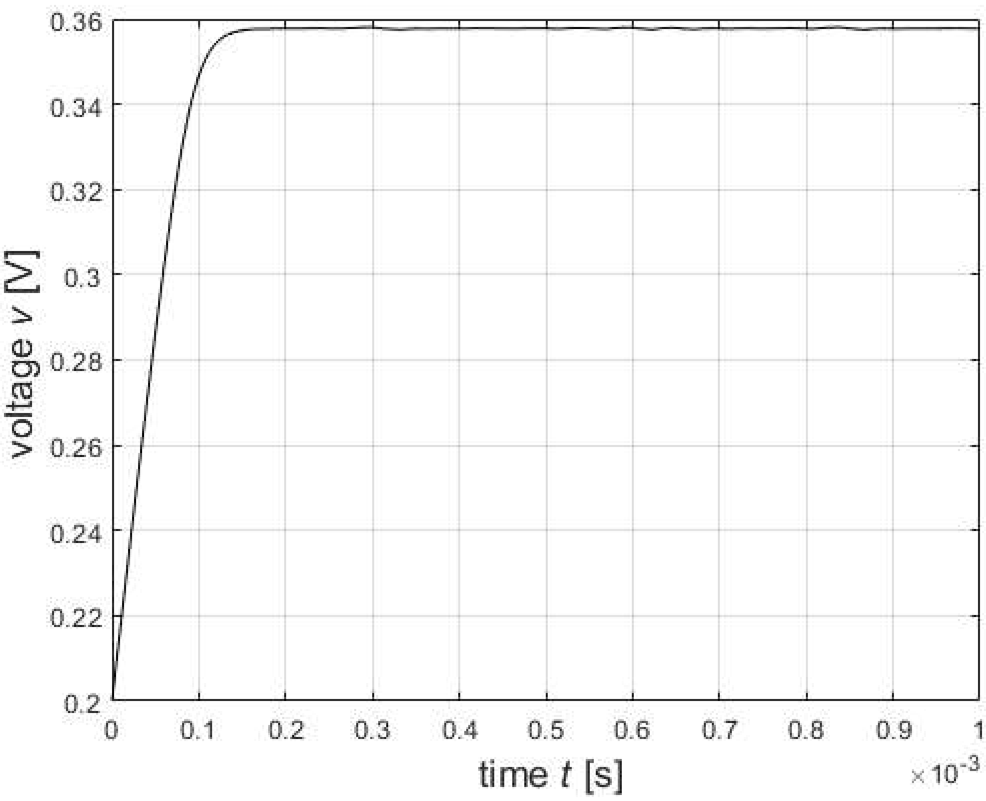
PUNTO DI EQUILIBRIO :

$$\frac{dv(t)}{dt}=0 \Rightarrow V_0 + RI_0\left(e^{\frac{V_0}{V_T}}-1\right) - V_S = 0$$

con $V_S(t) = V_S$

```
% ODE per circuito con diodo e condensatore
%
y0=0.2;
[t,y]=ode45(@Diodo_cond,[0 1e-3],y0);
plot(t,y,'k');
grid;
xlabel('{\fontsize{14} time \it{t} \rm[s]}');
ylabel('{\fontsize{14} voltage \it{v} \rm[V]}');


% Circuito con diodo e condensatore
function dy = Diodo_cond(t,y)
R = 1;
C = 1e-3;
I0 = 1e-6;
vs = 2;
dy = [-y/(R*C)-I0/C*exp(40*y)+I0/C+vs/(R*C)];
end
```



Published with MATLAB® R2022b

```
% Example of application of Newton method to find a zero of a circuit
% with a diode in parallel with a capacitor

x0 =  0.4; % make a starting guess at the solution
options = optimset('Display','iter','TolFun',10^(-9)); % option to display output
% Note the option TolFun: function tolerance
[x, fval] = fsolve(@Diodo_C_eq, x0, options); % call optimizer
sprintf('solution: v = %15.10f', x)

% the jacobian matrix is given in closed form
x0 = 0.4; % make a starting guess at the solution
disp('jacobian on');
options = optimset('Display','iter','Jacobian','on','TolFun',10^(-9)); % option to display output
[xj, fvalj] = fsolve(@Diodo_C_eq, x0, options); % call optimizer
sprintf('solution: v = %15.10f', xj)

% Function for Newton method of circuit for newton_diodoC1.m
function [F,J] = Diodo_C_eq(x)
F = x + 10^(-6)*exp(40*x)-10^(-6)-2;   % function evaluated at x
if nargout > 1   % two output arguments (J: Jacobian)
    J = 1 + 40*10^(-6)*exp(40*x)       % Jacobian of the function evaluated at x
end
end
```

```
                                        Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)          step         optimality    radius
     0          2         53.0874                       2.6e+03               1
     1          4         5.30159      0.0204411            364               1
     2          6        0.307339      0.0145804           49.1               1
     3          8      0.00399772     0.00625881           4.37               1
     4         10     1.26629e-06    0.000913952         0.0751               1
     5         12     1.39928e-13    1.68627e-05       2.49e-05               1
     6         14     2.36098e-26    5.60921e-09       1.02e-11               1

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.


ans =

    'solution: v =    0.3577888688'

jacobian on

J =

  356.4444


                                        Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)          step         optimality    radius
     0          1         53.0874                       2.6e+03               1
```

```
J =

  157.9184

     1          2         5.30158      0.0204411            364               1

J =

   88.5761

     2          3        0.307338      0.0145804           49.1               1

J =

   69.1803

     3          4      0.00399769     0.00625881           4.37               1

J =

   66.7328

     4          5     1.26623e-06    0.000913948         0.0751               1

J =

   66.6885

     5          6     1.39668e-13    1.68623e-05       2.49e-05               1

J =

   66.6885

     6          7     1.66923e-27    5.60399e-09       2.72e-12               1

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.


ans =

    'solution: v =    0.3577888688'
```