```
%% PROGETTO 2

I_s = 2e-6;
R = 1e3;
C = 3e-6;
T = 20e-3;

v_infty = R * I_s;
tau = R * C;
G = 1 / R;

v = @(t) v_infty .* (1 - exp(-t ./ tau));
i_R = @(t) G .* v(t);
i_C = @(t) I_s .* exp(-t ./ tau);

% 1. Derivazione Numerica

hvec = T./(2.^ [0:10]');
errvec = [];

for k = 1:length(hvec)
    h = hvec(k);

    tn = [0:h:T]';

    vn = v(tn);
    iCn = i_C(tn);

    N = length(tn);
    dvdt = zeros(N, 1);
    for t = 1:(N - 1)
        % in avanti per tutti tranne l'ultimo
        dvdt(t) = (vn(t + 1) - vn(t)) / h;
    end
    % all'indietro per l'ultimo
    dvdt(end) = (vn(end) - vn(end - 1)) / h;

    iCh = C * dvdt;

    en = iCn - iCh;

    errvec = [errvec; max(abs(en))];
end

%order_estimate(hvec, errvec);

t0 = T / 2;
tol = 1e-10;
fun = @(t) v_infty .* (1 - exp(-t ./ tau)) - 1e-3;
dfun = @(t) (v_infty ./  tau) .* exp(-t ./ tau);
[tstar_vect, k_new] = newton(t0, 1000, tol, fun, dfun);

% 2. Interpolazione Globale
```

```
a = 0;
b = T;

nvec = [1:10]';

err_vec = [];

xx = [a:(b-a)/10000:b]';

iCex = i_C(xx);

for i = 1:length(nvec)
    n = nvec(i);

    h = (b - a) / n;

    xi = [a:h:b]';

    fi = i_C(xi);

    coefficients = polyfit(xi, fi, n);
    Pn = polyval(coefficients, xx);

    er = iCex - Pn;

    err_max = max(abs(er));

    err_vec = [err_vec ; err_max];
end

figure(1);
semilogy(nvec, err_vec, "o");
xlabel("Grado del polinomio (n)");
ylabel("Errore max abs");
title("Errori di interpolazone");
grid on;

% 3. Interpolazione a Tratti

r = 2;

Mh_vec = 2.^ [1:10]';

err_vec = [];

H_vec = [];

for k = 1:length(Mh_vec)
    M = Mh_vec(k); % Num di sottointervalli
    h = (b - a) / M;
    H_vec = [H_vec; h];
```

```
    % Definisce i sottointervalli
    xi = linspace(a, b, M + 1)';

    Pn_vals = zeros(size(xx));

    % Per ogni sottointervallo
    for j = 1:M
        % Per interpolazione quadratica servono 2+1=3 nodi
        t_interp_nodes = [xi(j); (xi(j) + xi(j+1))/2; xi(j+1)];

        y_interp_vals = i_C(t_interp_nodes);

        % polin. interpolante di grado 2
        coefficients = polyfit(t_interp_nodes, y_interp_vals, r);

        % iindici su cui valutare il polinomio
        start_index = find(xx >= xi(j), 1, 'first');
        end_index = find(xx <= xi(j+1), 1, 'last');

        % valuta il polinomio solo nel sottointervallo attuale
        Pn_vals(start_index:end_index) = polyval( ...
            coefficients, xx(start_index:end_index));
    end
    er = max(abs(iCex - Pn_vals));
    err_vec = [err_vec ; er];
end

figure(2);
loglog(H_vec, err_vec, "-o");
xlabel("h");
ylabel("errore");

% order_estimate(H_vec, err_vec);
```