```
function [thetalist, success, theta_mat, err_ang, err_lin] = IKinBodyIterates(Blist, M,↙
T, thetalist0, max_iter, eomg, ev, filename)
% Takes Blist: The joint screw axes in the end-effector frame when the
%              manipulator is at the home position, in the format of a
%              matrix with the screw axes as the columns,
%       M: The home configuration of the end-effector,
%       T: The desired end-effector configuration Tsd,
%       thetalist0: An initial guess of joint angles that are close to
%                   satisfying Tsd,
%       max_iter: the maximum number iteration trying find the solution
%       eomg: A small positive tolerance on the end-effector orientation
%             error. The returned joint angles must give an end-effector
%             orientation error less than eomg,
%       ev: A small positive tolerance on the end-effector linear position
%           error. The returned joint angles must give an end-effector
%           position error less than ev.
%       filename: the csv filename to save the configuration
% Returns thetalist: Joint angles that achieve T within the specified
%                    tolerances,
%         success: A logical value where TRUE means that the function found
%                  a solution and FALSE means that it ran through the set
%                  number of maximum iterations without finding a solution
%                  within the tolerances eomg and ev.
%           theta_mat: the matrix of the all thetalist on all iterations
%           err_ang: list of angular errors
%           err_lin: list of linear errors
% Uses an iterative Newton-Raphson root-finding method.
% The maximum number of iterations before the algorithm is terminated has
% been hardcoded in as a variable called maxiterations. It is set to 20 at
% the start of the function, but can be changed if needed.
% Example Inputs:
%
% clear; clc;
% Blist = [[0; 0; -1; 2; 0; 0], [0; 0; 0; 0; 1; 0], [0; 0; 1; 0; 0; 0.1]];
% M = [[-1, 0, 0, 0]; [0, 1, 0, 6]; [0, 0, -1, 2]; [0, 0, 0, 1]];
% T = [[0, 1, 0, -5]; [1, 0, 0, 4]; [0, 0, -1, 1.6858]; [0, 0, 0, 1]];
% thetalist0 = [1.5; 2.5; 3];
% eomg = 0.01;
% ev = 0.001;
% max_iter = 30;
% filename = 'thetalist';
% [thetalist, success, theta_mat, err_ang, err_lin] = IKinBodyIterates(Blist, M, T,↙
thetalist0, max_iter, eomg, ev, filename)
%
% Outputs:
% --------------- Iteration 1 ---------------
% Joint Vector:
%     1.5000    2.5000    3.0000
%
% SE(3) end-effector config:
%    -0.0707    0.9975         0   -4.4887
%     0.9975    0.0707         0    4.3183
%          0         0   -1.0000    1.7000
%          0         0         0    1.0000
%
%           error twist V_b: (0.000, 0.000, 0.071, -0.300, -0.522, 0.014)
% angular error ||omega_b||: 7.079633e-02
%  linear error     ||v_b||: 6.025602e-01
%
% --------------- Iteration 2 ---------------
% Joint Vector:
%     1.5824    2.9748    3.1531
%
% SE(3) end-effector config:
%    -0.0001    1.0000         0   -4.9744
%     1.0000    0.0001         0    3.9423
%          0         0   -1.0000    1.6847
%          0         0         0    1.0000
%
%           error twist V_b: (0.000, 0.000, 0.000, 0.058, -0.026, -0.001)
% angular error ||omega_b||: 1.107873e-04
%  linear error     ||v_b||: 6.311800e-02
%
% --------------- Iteration 3 ---------------
% Joint Vector:
%     1.5707    2.9997    3.1415
%
% SE(3) end-effector config:
%     0.0000    1.0000         0   -4.9997
%     1.0000   -0.0000         0    4.0003
%          0         0   -1.0000    1.6858
%          0         0         0    1.0000
%
%           error twist V_b: (0.000, 0.000, -0.000, -0.000, -0.000, 0.000)
% angular error ||omega_b||: 4.608708e-06
%  linear error     ||v_b||: 4.444047e-04
%
% thetalist =
%     1.5707
%     2.9997
%     3.1415
% success =
%    1
% theta_mat =
%     1.5000    2.5000    3.0000
%     1.5824    2.9748    3.1531
%     1.5707    2.9997    3.1415
% err_ang =
%     0.0708
%     0.0001
%     0.0000
% err_lin =
%     0.6026
%     0.0631
%     0.0004


    theta_curr = thetalist0;
    theta_mat  = [];
    err_ang    = [];
    err_lin    = [];

    for i = 1 : max_iter
        fprintf('--------------- Iteration %d ---------------\n', i)

        theta_mat(i, :) = theta_curr';

        fprintf('Joint Vector:\n')
        disp(theta_curr')

%         calculate the transformation Tbd from body frame to desired frame
        Tsb = FKinBody(M, Blist, theta_curr);
        Tbd = TransInv(Tsb)*T;

        fprintf('SE(3) end-effector config:\n')
        disp(Tsb)

%         calculate the error twist vector
        Vb_skrw = MatrixLog6(Tbd);
        Vb_vec  = se3ToVec(Vb_skrw);

        wb = Vb_vec(1:3, :);
        vb = Vb_vec(4:6, :);

        fprintf('          error twist V_b: ')

        fprintf('(')
        for j = 1:6
            if j == 6
                fprintf('%.3f', Vb_vec(j))
            else
                fprintf('%.3f, ', Vb_vec(j))
            end
        end
        fprintf(')\n')

%         calculate the error
        err_ang(i, 1) = norm(wb);
        err_lin(i, 1) = norm(vb);

        fprintf('angular error ||omega_b||: %d\n', norm(wb))
        fprintf(' linear error     ||v_b||: %d\n\n', norm(vb))
```

```
%         return the result if converges
        if norm(wb) < eomg && norm(vb) < ev
            thetalist = theta_curr;
            success   = true;
            writematrix(theta_mat, sprintf('%s.csv', filename));
            return
        end

%         calculate the psuedo jacobian and the new theta
        J   = JacobianBody(Blist, theta_curr);
        J_p = pinv(J);
        dtheta = J_p*Vb_vec;

        theta_curr = theta_curr + dtheta;
        theta_curr = atan2(sin(theta_curr), cos(theta_curr));
    end

%     Fail if not concerge after all iterations
    thetalist = NaN;
    success = false;
end
```