```
% Tests for LS-GA, LS, l1, P1, P2(1) methods without noise

% clear workspace and close all figures
close all;
clearvars;

% Setting parameters:
k = 16; % Number of sensors
m = 4; % Size of observation vectors b
n = 20; % Size of unknown vector x
reliable_sensors_list = [6 8 10 12 14]; % Number of consistent↙
sensors
delta = 1e-6; % Concave approximation related constant
threshold = 1e-4; % Threshold to recover reliable sensors
MCexperiments = 1000;
methods = 5; % Methods being studied ( LS-GA, LS, l1, P1, P2(1) )

% save results
results = zeros(methods, length(reliable_sensors_list));

for s_index = 1:length(reliable_sensors_list)

    s = reliable_sensors_list(s_index);
    reliable_sensors = [ones(1, s) zeros(1, k-s)];

    fprintf('Considered %d reliable sensors. ', s);

    parfor j=1:MCexperiments

        %preallocations
        bi = zeros(m, 1, k);

        % unknown vector is modeled as x0 ~ N(0, n^(-1/2)In)
        x0 = mvnrnd(zeros(1, n), n^(-1)*eye(n))';

        % Entries of matrix A are drawn independently from N(0, 1)
        Ai = randn(m, n, k);

        % generate sensor observation data b
        % consistent observations
        for i=1:s
            bi(:, :, i) = Ai(:, : ,i)*x0;
        end
        % unreliable sensors
        for i=s+1:k
            bi(:, : , i) = mvnrnd(zeros(1, m), eye(m))';
        end

        % Rearrange arrays and matrices
        b = bi(:);
        C = permute(Ai, [1 3 2]);
        A = reshape(C, [], size(Ai, 2), 1);
        b_ga = b(1:m*s);
        A_ga = A(1:m*s,:)

        % LS-GA method
        x_ls_ga = ls_method(A_ga, b_ga, n);
        sensor_results_ls_ga = sensor_validation(Ai, bi, x_ls_ga,↙
threshold, k, s);
        results_ls_ga(j, s_index) = isequal(reliable_sensors,↙
sensor_results_ls_ga);

        % LS method
        x_ls = ls_method(A, b, n);
        sensor_results_ls = sensor_validation(Ai, bi, x_ls,↙
threshold, k, s);
        results_ls(j, s_index) = isequal(reliable_sensors,↙
sensor_results_ls);

        % l1 method
        x_l1 = l1_method(A, b, n);
        sensor_results_l1 = sensor_validation(Ai, bi, x_l1,↙
threshold, k, s);
        results_l1(j, s_index) = isequal(reliable_sensors,↙
sensor_results_l1);

        % P1 method
        x_p1 = p1_method(Ai, bi, n, k);
        sensor_results_p1 = sensor_validation(Ai, bi, x_p1,↙
threshold, k, s);
        results_p1(j, s_index) = isequal(reliable_sensors,↙
sensor_results_p1);

        % P2(1) method
        x_p2_1 = p2_1_method(Ai, bi, n, k, x_p1, delta);
        sensor_results_p2_1 = sensor_validation(Ai, bi, x_p2_1,↙
threshold, k, s);
        results_p2_1(j, s_index) = isequal(reliable_sensors,↙
sensor_results_p2_1);
    end
end

results(1,:) = sum(results_ls_ga, 1);
results(2,:) = sum(results_ls, 1);
results(3,:) = sum(results_l1, 1);
results(4,:) = sum(results_p1, 1);
results(5,:) = sum(results_p2_1, 1);
results = (results./MCexperiments).*100;

% Print results
f = figure('Position',[440 500 500 140]);
% Create the column and row names in cell arrays
cnames = {'s=6','s=8','s=10', 's=12', 's=14'};
rnames = {'LS-GA', 'LS','L1','P1', 'P2(1)'};

% Create the uitable
t = uitable(f,'Data',results,...
            'ColumnName',cnames,...
            'RowName',rnames);

% Set width and height
t.Position(3) = t.Extent(3);
t.Position(4) = t.Extent(4);
```