```
% 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);
```

```
% Tests for LS-GA, LS, l1, P1, P2(1) methods with 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
s = 14; % Number of consistent sensors
delta = 1e-6; % Concave approximation related constant
methods = 4; % Methods being studied (LS, l1, P1, P2(1) )
SNR = [5 10 15 20 25]; % SNR wanted
noise_levels_sigma = (10.^(-SNR/20));
MCexperiments = 1000;

% save all MSE values for all methods
results_mse = zeros(length(noise_levels_sigma), methods);

for noise_index = 1:length(noise_levels_sigma)

    noise_sigma = noise_levels_sigma(noise_index);

    fprintf('Considered SNR: %d. ', SNR(noise_index));

    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);

        for i=1:s
            % reliable sensors measures
```

```
            vi = mvnrnd(zeros(1, m), (noise_sigma^2)*eye(m))';
            bi(:, :, i) = Ai(:, : ,i)*x0 + vi;
        end

        for i=s+1:k
            % unreliable sensors measures
            bi(:, : , i) = mvnrnd(zeros(1, m), (1+noise_sigma^2)↙
*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);
        results_noise_ls_ga(j, noise_index) = norm(x0-x_ls_ga)^2;

        % LS method
        x_ls = ls_method(A, b, n);
        results_noise_ls(j, noise_index) = norm(x0-x_ls)^2;

        % l1 method
        x_l1 = l1_method(A, b, n);
        results_noise_l1(j, noise_index) = norm(x0-x_l1)^2;

        % P1 method
        x_p1 = p1_method(Ai, bi, n, k);
        results_noise_p1(j, noise_index) = norm(x0-x_p1)^2;

        % P2(1) method
        x_p2_1 = p2_1_method(Ai, bi, n, k, x_p1, delta);
        results_noise_p2_1(j, noise_index) = norm(x0-x_p2_1)^2;
    end
end
```

```
results_mse(:,1) = mean(results_noise_ls_ga, 1);
results_mse(:,2) = mean(results_noise_ls, 1);
results_mse(:,3) = mean(results_noise_l1, 1);
results_mse(:,4) = mean(results_noise_p1, 1);
results_mse(:,5) = mean(results_noise_p2_1, 1);

% plot data and add pretty stuff
semilogy(results_mse, '.-', 'MarkerSize',20, 'LineWidth', 1.5)
title('MSE variation with SNR')
xlabel('SNR [dB]')
ylabel('MSE')
legend('LS-GA', 'LS', 'L_1', 'P_1', 'P_2(1)', 'Location',↙
'southwest');
ax = gca;
ax.XTick = [1 2 3 4 5];
ax.XTickLabel = SNR;
grid on;
```

```
% Least squares method, cvx aided
function [ x ] = ls_method( A, b, n )
    cvx_begin quiet
        variable x(n, 1)
        minimize( (b-A*x)'*(b-A*x) )
    cvx_end
end
```

```
% Least squares method (norm 1), cvx aided
function [ x ] = l1_method( A, b, n )
    cvx_begin quiet
        variable x(n, 1)
        minimize( norm(b-A*x, 1) )
    cvx_end
end
```

```
% convex relaxation method, cvx aided
function [ x ] = p1_method( A, b, n, k )
    cvx_begin quiet
        variable x(n, 1)
        % define cost function
        f = 0;
        for i=1:k
            f = f + norm( b(:, :, i) - A(:, :, i)*x );
        end
        minimize(f)
    cvx_end
end
```

```
% concave approximation method, cvx aided
function [ x ] = p2_1_method( A, b, n, k, x0, delta )
    for i=1:k
       w(i) = ( norm( b(:,:,i)-A(:,:,i)*x0 ) + delta )^(-1);
    end
    cvx_begin quiet
        variable x(n, 1)
        % define cost function
        f = 0;
        for i=1:k
            f = f + w(i)*norm( b(:,:,i)-A(:,:,i)*x );
        end
        minimize(f)
    cvx_end
end
```

```
% Sensor validation
function [ results ] = sensor_validation( A, b, x, threshold, k, s)
results = zeros(1, k);
for i=1:s
    %norm inf must be lower than the threshold for all reliable
sensors
    results(i) = norm(b(:,:,i) - A(:,:, i)*x, inf) <= threshold;
end
end
```

```
% Tests for Matching Solutions method without noise with 2↙
iterations

close all;
clearvars;

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
restriction_delta = 10^-10;
threshold = 10^-4;
MCexperiments = 1000;

for s_index = 1:length(reliable_sensors_list)

    s = reliable_sensors_list(s_index);

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

    reliable_sensors = [ones(1, s) zeros(1, k-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

        x_est_0 = randn(n, k) / sqrt(n);
        lambda_est_0 = mean(x_est_0, 2);

        % lambda_est = matching_solutions( Ai, bi, n, k,↙
restriction_delta, x0, L0);
        [x_est_1, lambda_est_1] = matching_solutions_miter( Ai, bi,↙
n, k, restriction_delta, x_est_0, lambda_est_0);
        [x_est_2, lambda_est_2] = matching_solutions_miter( Ai, bi,↙
n, k, restriction_delta, x_est_1, lambda_est_1);

        % check for reliable sensors
        method_reliable_sensors_iter1 = sensor_validation( Ai, bi,↙
lambda_est_1, threshold, k, s);
        method_reliable_sensors_iter2 = sensor_validation( Ai, bi,↙
lambda_est_2, threshold, k, s);

        list_results_iter1(j, s_index) = isequal(reliable_sensors,↙
method_reliable_sensors_iter1);
        list_results_iter2(j, s_index) = isequal(reliable_sensors,↙
method_reliable_sensors_iter2);
    end
end

results(1, :) = sum(list_results_iter1, 1);
results(2, :) = sum(list_results_iter2, 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 = {'MS(1)', 'MS(2)'};
```

```
% 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);
```

```
% Tests for Matching Solutions method with noise

close all;
clearvars;

k = 16; % Number of sensors
m = 4; % Size of observation vectors b
n = 20; % Size of unknown vector x
s = 14;
SNR = [5 10 15 20 25]; % SNR wanted
noise_levels_sigma = (10.^(-SNR/20));
restriction_delta = 10^-10;
threshold = 10^-4;
MCexperiments = 1000;

for noise_index = 1:length(noise_levels_sigma)

    noise_sigma = noise_levels_sigma(noise_index);
    reliable_sensors = [ones(1, s) zeros(1, k-s)];

    fprintf('Considered SNR: %d. ', SNR(noise_index));

    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);

        for i=1:s
            % reliable sensors measures
            vi = mvnrnd(zeros(1, m), (noise_sigma^2)*eye(m))';
            bi(:, :, i) = Ai(:, : ,i)*x0 + vi;
        end

        for i=s+1:k
```

```
            % unreliable sensors measures
            bi(:, : , i) = mvnrnd(zeros(1, m), (1+noise_sigma^2)↙
*eye(m))';
        end

        x_iter0 = randn(n, k) / sqrt(n);
        lambda_iter0 = mean(x_iter0, 2);

        lambda_iter1 = matching_solutions( Ai, bi, n, k,↙
restriction_delta, x_iter0, lambda_iter0);
        results_noise_ms(j, noise_index) = norm(x0-lambda_iter1)^2;
    end
end

results_mse = mean(results_noise_ms, 1);

% plot data and add pretty stuff
semilogy(results_mse, '.-', 'MarkerSize',20, 'LineWidth', 1.5)
title('MSE variation with SNR')
xlabel('SNR [dB]')
ylabel('MSE')
legend('MS(1)', 'Location', 'southwest');
ax = gca;
ax.XTick = [1 2 3 4 5];
ax.XTickLabel = SNR;
grid on;
```

```
% Matching solutions method
function [ x, L ] = matching_solutions_miter( A, b, n, k, delta,
x0, L0)

    cvx_begin quiet
        variable x(n, k)
        variable L(n, 1)
        % define cost function
        for i=1:k
            f(i) =  norm(x(:,i)-L) / norm(x0(:,i)-L0);
        end

        minimize(sum(f))

        subject to
        for i=1:k
            (b(:,:,i)-A(:,:,i)*x(:,i))'*(b(:,:,i)-A(:,:,i)*x(:,i))
<= delta;
        end
    cvx_end
end
```