Contents

```
clear; close all; clc;
```

a)

There are two depth cues for the distance to the beach. Both of the cues are independent Gaussian random variables. These are unbiased cues implies that the expected value of the cue is equal to the actual value of the depth.

b)

Let the depth estimate due to the first cue be D_1 and the depth estimate due to the second cue be D_2. Let the weightage given to the first cue be w. Therefore, the weightage given to the second cue will be 1-w. Hence, the combined estimate of depth becomes:

$$D = wD_1 + (1-w)D_2$$

Therefore, the variance of the estimate will be:

$$Var(D) = Var(wD_1 + (1-w)D_2) = Var(wD_1) + Var((1-w)D_2)$$

Therefore,

$$Var(D) = w^2Var(D_1) + (1-w)^2Var(D_2)$$

An optimal observer would try to minimize the resulting estimate of the variance of depth. Therefore, the goal of the optimal observer would be to make the derivative of Var(D) = 0.

$$\frac{dVar(D)}{dw} = 2wVar(D_1) - 2(1-w)Var(D_2) = 0$$

Solving for w:

$$wVar(D_1) - (1-w)Var(D_2) = 0$$

Therefore,

$$wVar(D_1) - Var(D_2) + wVar(D_2) = 0$$

Therefore,

$$w(Var(D_1) + Var(D_2)) = Var(D_2)$$

Therefore,

$$w = \frac{Var(D_2)}{Var(D_1) + Var(D_2)}$$

We can check by taking the second-derivative of V that this is indeed the w that minimizes the variance of the estimate.

The given variances of the two cues are:

$$Var(D_1) = 4, Var(D_2) = 1$$

Substituting back into the formula for w, we get:

$$w = \frac{1}{4+1} = \frac{1}{5} = 0.2$$

Also,

$$1 - w = 1 - \frac{1}{5} = \frac{4}{5} = 0.8$$

Therefore, an optimal observer would give a weight of 0.2 to the first cue and a weight of 0.8 to the second cue. Plugging the weights back into the formula for computing the variance of the output, we get:

$$Var(D) = 0.2^2 \times 4 + 0.8^2 \times 1$$

Therefore,

$$Var(D) = 0.04 \times 4 + 0.64 \times 1$$

Therefore,

$$Var(D) = 0.16 + 0.64$$

Therefore,

$$Var(D) = 0.80$$

c)

If, instead, equal weigths were given to the two cues, then we have:

$$w = 1 - w = 0.5$$

Therefore, the estimates of the variance of the depth now becomes:

$$Var(D) = 0.5^2 \times 4 + 0.5^2 \times 1$$

Therefore,

$$Var(D) = 0.25 \times 4 + 0.25 \times 1$$

Therefore,

$$Var(D) = 1 + 0.25$$

Therefore,

$$Var(D) = 1.25$$

The difference between the variance of the optimal estimator and this estimator is 1.25 - 0.8 = 0.45

d)

As can be seen, the "average cue" has a higher variance than the second cue. Therefore, David would be better off selecting just the second cue and ignoring the first cue instead of averaging the two cues.

Contents

```
clear; close all; clc;
```

This problem extends problem 1 to a 2-dimensional case. The target is a circle of radius = 1. There are two cues both of which are bivariate Gaussians with different variances. Since these are unbiased cues, their mean estimates are the actual position of the target center. The task here is to construct bivariate Gaussians for cue1 (using Gaussians X1 and Y1) and for cue2 (using Gaussians X2 and Y2). Given that X1 and Y1 are independent and the same is the case for X2 and Y2, the bivariate Gaussian samples can be simply obtained by drawing samples from each 1-D Gaussian independently. The sampling 1-D Gaussians can be done using randn function of MATLAB. randn functions draws samples from a 1-D Gaussian of standard deviation = 1 and mean = 0. The standard deviation of the Gaussian can be changed by multiplying the output sample by the standard deviation of choice. Additionally, the mean can be incorporated by translating the output by the desired mean.

```
target_radius = 1; % radius of the target
x_center = randi(10); % The x-coordinate of the center of the target
y_center = randi(10); % The y-coordinate of the center of the target

var1 = 1; % variance of both Gaussians for cue1
var2 = 4; % variance of both Gaussians for cue2

trials = 1e6; % Number of trials to be run for the Monte Carlo simulation

% Drawing samples from Gaussians
X1 = x_center + sqrt(var1) .* randn(trials, 1);
X2 = x_center + sqrt(var2) .* randn(trials, 1);
Y1 = y_center + sqrt(var1) .* randn(trials, 1);
Y2 = y_center + sqrt(var2) .* randn(trials, 1);
```

a)

If Michael uses only cue 1, then the samples drawn from the Gaussian X1 and Y1 are the x and y coordinates of the estimates of the center of the target. The estimate is correct if it lies inside the circle of radius 1 around the center. The radial distance of each estimate from the center can be computed using the Euclidean distance formula:

$$d = \sqrt{(x-\bar{x})^2 + (y-\bar{y})^2}$$

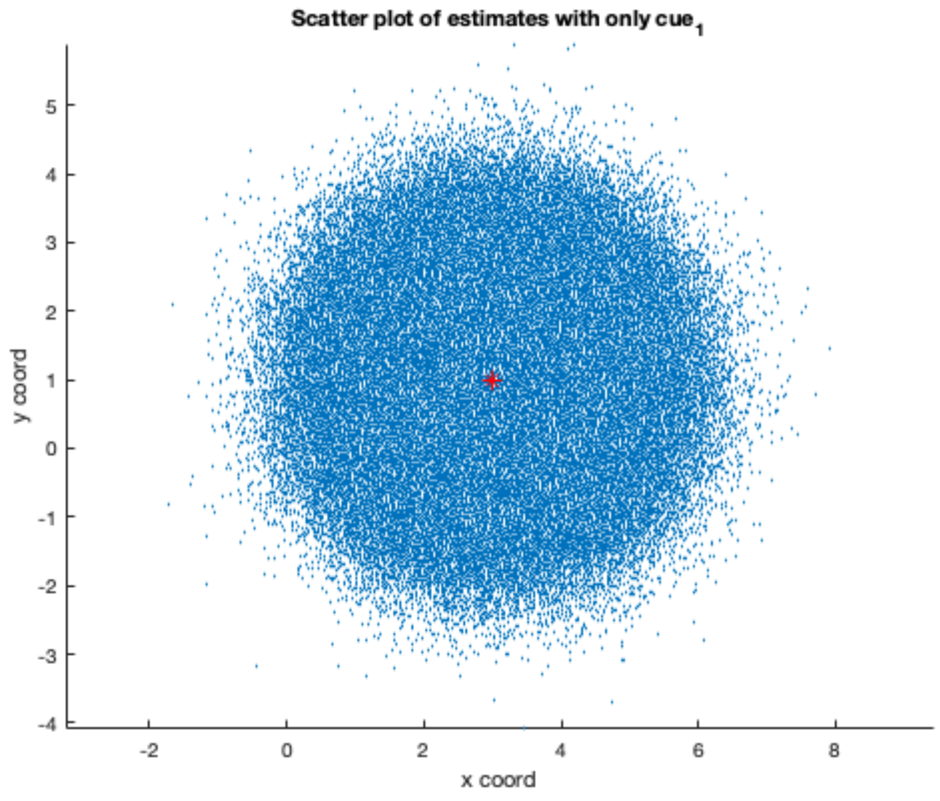
And the estimate is correct if this radial distance is less than or equal to the radius of the circle which is 1.

The probability of correct estimate can then be computed as the ratio of the count of the correct estimates by the total number of trials.

```
figure();
scatter(X1, Y1, 1)
hold on;
plot(x_center, y_center, 'r*')
xlabel('x coord')
ylabel('y coord')
title('Scatter plot of estimates with only cue_1')
axis equal
```

```
radial_distance = sqrt((X1 - x_center).^2 + (Y1 - y_center).^2);
count_correct = sum(radial_distance <= target_radius);
prob_correct = count_correct/trials
```

```
prob_correct =

    0.3936
```



b)

If instead, Michael were to use both the cues, then he combines X1, X2 and Y1, Y2 to compute the estimate X, Y. The weighting used to compute X is:

$X = wX_1 + (1 - w)X_2$

Similarly, the weighting used to compute Y is:

$Y = vY_1 + (1 - v)Y_2$

Assuming that Michael is making an optimal choice, then the goal would be to minimize the variance of the estimates X and Y. From Q1, we know that when the variances are minimized, the weights are given by:

$$w = \frac{var(X_2)}{var(X_1) + var(X_2)}$$

Similarly,

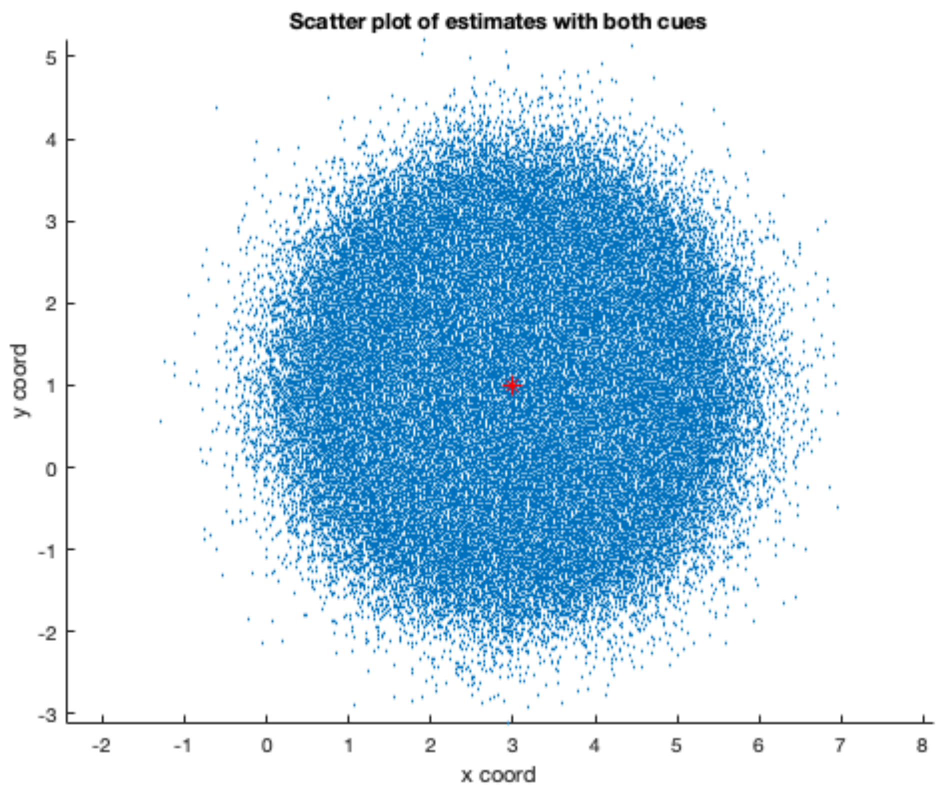
$$v = \frac{var(Y_2)}{var(Y_1) + var(Y_2)}$$

These weights can then be used to compute the X and Y estimates of the center.

Following the procedure in (a), the correct estimates can then be computed by checking if the radial estimate of each of these estimates from the center of the target circle is within radius of 1. The probability correct is then the ratio of the count of the correct estimates and the total number of trials.

```
w = var2/(var1 + var2);
v = var2/(var1 + var2);

X = w * X1 + (1 - w) * X2;
Y = v .* Y1 + (1 - v) .* Y2;

figure();
scatter(X, Y, 1)
hold on;
plot(x_center, y_center, 'r*')
xlabel('x coord')
ylabel('y coord')
title('Scatter plot of estimates with both cues')
axis equal

radial_distance = sqrt((X - x_center).^2 + (Y - y_center).^2);
count_correct = sum(radial_distance <= target_radius);
prob_correct = count_correct/trials
```

```
prob_correct =

    0.4656
```



From the simulation, we see that weighing the two cues optimally would increase the probability of Michael hitting the target correctly.

Published with MATLAB® R2021b

Contents

```
clear; close all; clc;
```

a)

Denis is present in a one-dimensional shopping mail. Let the location of Denis be given by X. The prior of Denis' location is a Gaussian distribution with mean of 50 and a variance of 64. The posterior distribution is computed based on the location of coffee cup. The likelihood of the cup being at the location given Denis' position follows a Gaussian distribution with mean 30 and a variance of 100. The goal is to compute the posterior distribution.

The prior is p(X). The likelihood is p(C|X) and the posterior is p(X|C). Using Bayes' rule, the posterior can be computed as:

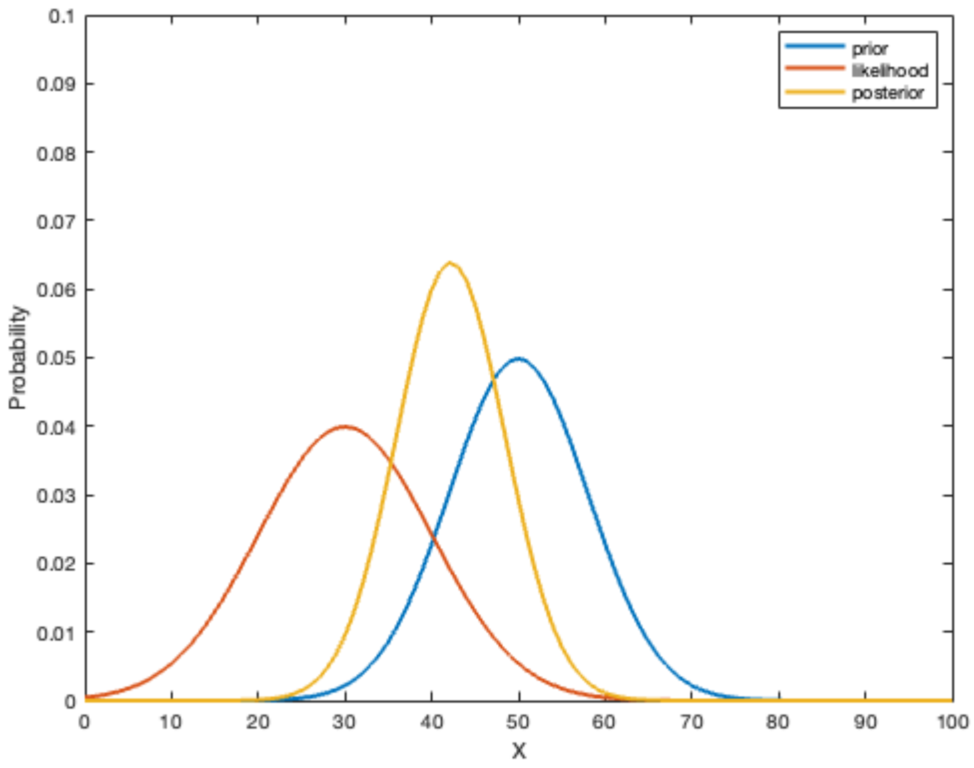
$$p(X|C) = \frac{p(C|X)p(X)}{p(C)}$$

Here p(C) is a normalizing factor and hence can be ignored for our purposes. Therefore, we have:

$$p(X|C) \sim p(C|X)p(X)$$

Using this formula, we can compute the posterior given the likelihood and prior distributions.

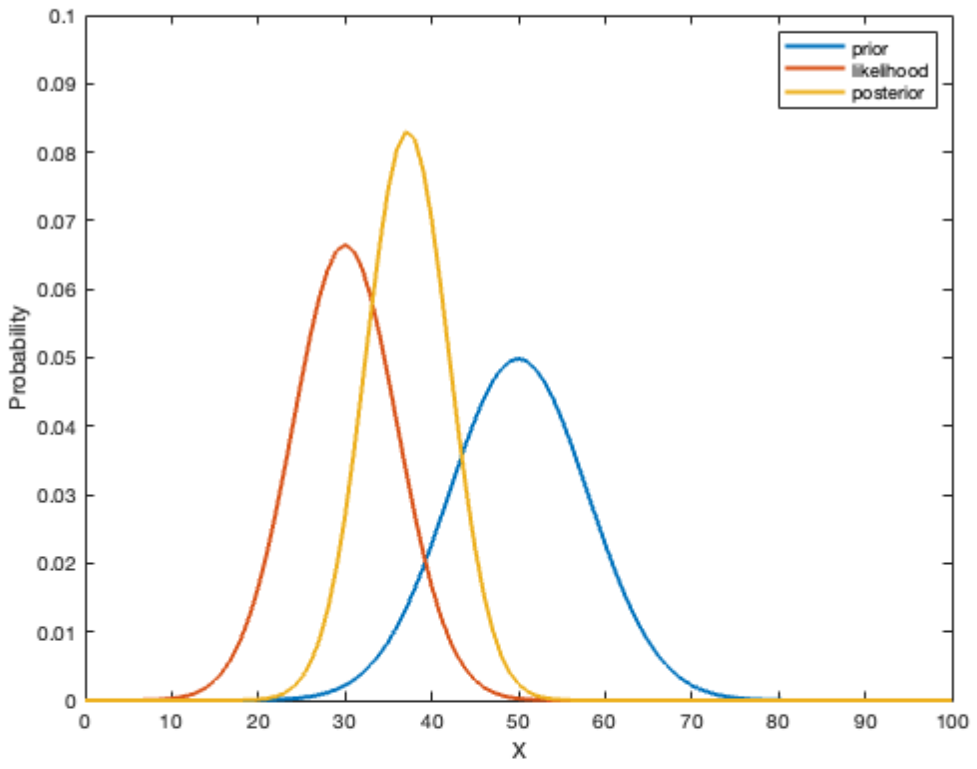
```
X = 0:100; % possible locations of Denis

% prior
Denis_prior_mean = 50;
Denis_prior_variance = 64;
prior_unnormalized = normpdf(X, Denis_prior_mean, sqrt(Denis_prior_variance));
prior = prior_unnormalized ./ sum(prior_unnormalized);

% likelihood
likelihood_mean = 30;
likelihood_variance = 100;
likelihood_unnormalized = normpdf(X, likelihood_mean, sqrt(likelihood_variance));
likelihood = likelihood_unnormalized ./ sum(likelihood_unnormalized);

% posterior
posterior_unnormalized = prior .* likelihood;
posterior = posterior_unnormalized ./ sum(posterior_unnormalized);

figure();
plot(X, prior, 'DisplayName', 'prior', 'LineWidth', 2)
hold on;
plot(X, likelihood, 'DisplayName', 'likelihood', 'LineWidth', 2)
plot(X, posterior, 'DisplayName', 'posterior', 'LineWidth', 2)
xlabel('X')
ylabel('Probability')
ylim([0, 0.1])
legend()
```



b)

In the scenario where the coffee cup is not that cold, the variance of the likelihood Gaussian decreases. However, the process of computing the posterior from the likelihood and prior remains the same.

```
X = 0:100; % possible locations of Denis

% prior
Denis_prior_mean = 50;
Denis_prior_variance = 64;
prior_unnormalized = normpdf(X, Denis_prior_mean, sqrt(Denis_prior_variance));
prior = prior_unnormalized ./ sum(prior_unnormalized);

% likelihood
likelihood_mean = 30;
likelihood_variance = 36;
likelihood_unnormalized = normpdf(X, likelihood_mean, sqrt(likelihood_variance));
likelihood = likelihood_unnormalized ./ sum(likelihood_unnormalized);

% posterior
posterior_unnormalized = prior .* likelihood;
posterior = posterior_unnormalized ./ sum(posterior_unnormalized);

figure();
plot(X, prior, 'DisplayName', 'prior', 'LineWidth', 2)
hold on;
plot(X, likelihood, 'DisplayName', 'likelihood', 'LineWidth', 2)
plot(X, posterior, 'DisplayName', 'posterior', 'LineWidth', 2)
xlabel('X')
ylabel('Probability')
ylim([0, 0.1])
legend()
```



We can see that as the mean of the likelihood than the mean of the prior, the likelihood pulls the posterior to have a lower mean in (a). The pull is also determined the variance of the likelihood. If the variance of the likelihood is high, then the certainty of the mean is lower and hence its impact on the pull of the posterior is weaker. On the other hand, if the variance is lower then the certainty of the mean is higher and hence its impact on the pull of the posterior is stronger. Therefore, compared to (a), we can see that the posterior in (b) has a lower mean and also a narrower distribution.

Published with MATLAB® R2021b

```
clear; close all; clc;
```

This problem is identical to the Q2 (b) with multiple cues and each cue having the same variance of 10. Using the same rationale, we can obtain the weight of each cue to be:

$$w = \frac{var(X_i)}{\sum_i var(X_i)}$$

Similarly,

$$v = \frac{var(X_i)}{\sum_i var(X_i)}$$

Since the variance of each cue is the same, we get:

$$w = \frac{1}{num\ of\ cues}$$

Similarly,

$$v = \frac{1}{num\ of\ cues}$$

These weights can then be used to compute the X and Y estimates of the center.

Following the procedure in 2a and 2b, the correct estimates can then be computed by checking if the radial estimate of each of these estimates from the center of the target circle is within radius of 1. The probability correct is then the ratio of the count of the correct estimates and the total number of trials.

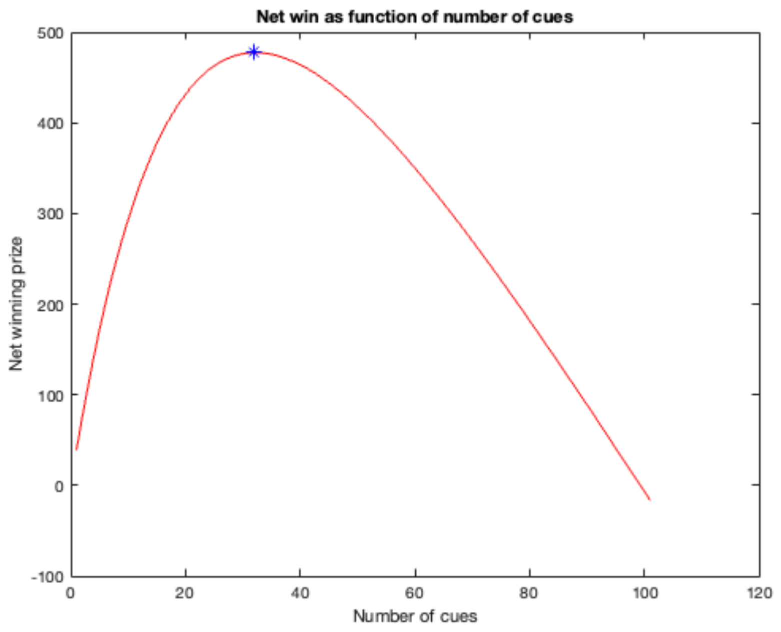
```
trials = 1e6; % Number of trials to be run for the Monte Carlo simulation
x_center = ones(trials, 1) .* randi(10); % The x-coordinate of the center of the target
y_center = ones(trials, 1) .* randi(10); % The y-coordinate of the center of the target
gauss_var = 10 .* ones(trials, 1); % variance of Gaussians for each cue
target_radius = ones(trials, 1); % radius of the target
cue_counts = 1:101; % number of cues used

% Drawing samples from Gaussians
X_cue = x_center + sqrt(gauss_var) .* randn(trials, length(cue_counts));
Y_cue = y_center + sqrt(gauss_var) .* randn(trials, length(cue_counts));

X = zeros(trials, length(cue_counts)); % Initializing x choices
Y = zeros(trials, length(cue_counts)); % Initializing y choices

for cc = cue_counts
    % Weights of cues
    w = ones(cc, 1)./cc;
    v = ones(cc, 1)./cc;
    % Final choices
    X(:, cc) = X_cue(:, 1:cc) * w;
    Y(:, cc) = Y_cue(:, 1:cc) * v;
end

radial_distance = sqrt((X - x_center).^2 + (Y - y_center).^2); % computing radial distance
count_correct = sum(radial_distance <= target_radius); % checking for correct hits
prob_correct = count_correct./trials; % probability of correct choices
```

Now the potential prize that the subject can win is $1000. The expected winning prize can hence be computed as the product of probability correct and the total winning prize. The cost of each cue is $10. Therefore, the total cost of cue is the product of total number of cues used and the cost of each cue. The net winning prize is then simply a difference between the total prize and the cue cost. The maximum winning prize can be obtained using the max function.

```
potential_prize = 1000;
cue_cost = 10;
prize = potential_prize .* prob_correct;
cue_cost = cue_cost .* cue_counts;

net_winning = prize - cue_cost;
[max_winning, max_win_index] = max(net_winning);

sprintf('In order to maximize the gain, the simulation suggests that Roozbeh should use approximately %d cues', max_win_index)
```

```
figure()
plot(cue_counts, net_winning, 'r-')
hold on;
plot(cue_counts(max_win_index), max_winning, 'b*')
xlabel('Number of cues')
ylabel('Net winning prize')
title('Net win as function of number of cues')
```

```
ans =

    'In order to maximize the gain, the simulation suggests that Roozbeh should use approximately 32 cues'
```



Published with MATLAB® R2021b