Exercise – Understanding rejection sampling using a Gaussian-oscillatory

1. Aim of the exercise

The aim of this exercise is to develop intuition for rejection sampling. Specifically, we use a Gaussian envelope combined with an oscillatory component as the target function, and apply rejection sampling to estimate the area under this function. By comparing accepted and rejected samples under different proposal distributions, the exercise illustrates how the choice of proposal affects efficiency and accuracy.

2. Theory

Before proceeding with the example, review the theoretical foundation of rejection sampling as developed in the dedicated theory exercise. The present exercise focuses on applying those principles to a concrete target function.

3. Clear the memory

Remove all existing variables from the workspace to ensure a clean computational environment.

```
%% 3. Clear the memory

% Clear the memory
clear;
```

4. Define a domain of values

We construct a fine grid of points over the interval $[-4, 4]$. The wider range is chosen for visualization, so that plots display the target function on its effective support $[-3, 3]$ together with the surrounding area.

```
%% 4. Define a domain of values

% 4.1. Specify number of random samples to estimate the integral
N_samples = 1000;

% 4.2. Define a domain of values
x = -4:0.001:4;
```

5. Construct the target function

Suppose we have the following unnormalized density

$$\tilde{f}_X(x) = e^{-\frac{1}{2}x^2} \cdot \left[\pi \cos^2(x) \cdot \sin^2(\pi x) + 1\right] \cdot \mathbf{1}_{[-3,3]}(x)$$

supported on $[-3, 3] \subseteq \mathbb{R}$. We code this function in three steps. First, we define the oscillatory component:

$$\pi \cos^2(x) \cdot \sin^2(\pi x) + 1.$$

Second, we preallocate a vector of the same length as the domain to store values of $\tilde{f}_X(x)$. Finally, we create a loop that assigns values piecewise: for x inside $[-3, 3]$, the Gaussian envelope $e^{-\frac{1}{2}x^2}$ is multiplied by the oscillatory component; for x outside this interval, the function is set to zero. This procedure enforces the indicator function $\mathbf{1}_{[-3,3]}(x)$ in the definition above.

```
%% 5. Construct the target function

% 5.1. Define the oscillatory component
oscillatory_component = pi*cos(x).^2.*sin(pi*x).^2 + 1;

% 5.2. Preallocate the target function vector
target_function = NaN(1,length(x));

% 5.3. Define the target function piecewise
for i = 1:length(x)
    % Check if x(i) lies within the domain [-3, 3]
    if abs(x(i)) <= 3
        % Inside domain: compute Gaussian * Oscillatory component
        target_function(i) = exp(-0.5*x(i).^2).* ...
            oscillatory_component(i);
    else
        % Outside domain: set target function to zero
        target_function(i) = 0;
    end
end
```

6. Construct a proposal distribution

The idea of rejection sampling is to choose a proposal density p_X such that the envelope defined by p_X fully covers the unnormalized density $\tilde{f}_X$, i.e.

$$\{(x, y) \in \mathbb{R}^2 : 0 \leq y \leq \tilde{f}_X(x)\} \subseteq \{(x, y) \in \mathbb{R}^2 : 0 \leq y \leq p_X(x)\}.$$

The simplest choice for a proposal density is the uniform distribution over the support of $\tilde{f}_X$, that is, $p_X \sim \text{Unif}(-3, 3)$. However, this proposal does not dominate $\tilde{f}_X$ everywhere. To ensure coverage, the proposal must be scaled by a constant C:

$$C := \sup\{f(x) \mid x \in \mathbb{R}\} \cdot (\max\{\text{Supp}(f)\} - \min\{\text{Supp}(f)\})$$

where

$$c := \sup\{f(x) \mid x \in \mathbb{R}\}.$$

We require that the scaled uniform proposal density attains c, the maximal value of f, on its support. The second factor in C accounts for the width of the support and serves as a regularization constant.

```
%% 6. Construct a proposal distribution

% 6.1. Specify the proposal distribution
proposal_uniform_PD = makedist('Uniform',-3,3);

```

```
% 6.2. Compute the proposal PDF
proposal_uniform_PDF = pdf(proposal_uniform_PD,x);

% 6.3. Determine the maximum of the target function
c = max(target_function);

% 6.4. Choose envelope constant
C = c*(3-(-3));
```

7. Visualize the target and proposal distributions

Figure 1 displays the target function $\tilde{f}_X$ together with the scaled uniform proposal density $C \cdot p_X \sim C \cdot \text{Unif}(-3, 3)$. The scaled proposal serves as an envelope for rejection sampling, ensuring that $\tilde{f}_X(x) \leq C \cdot p_X(x)$ for all x in the support. To illustrate, consider the point $x = -1.2$: here $\tilde{f}_X(-1.2)$ is much smaller than $C \cdot p_X(-1.2)$, showing how the envelope dominates the target function across its domain.

In the MATLAB code we produce this visualization. First, we create a new figure. We plot the target function in blue, the proposal PDF in default color, and the scaled proposal $C \cdot p_X$ in black. We mark the maximum of the target function with a dashed line, and highlight the reference point $x = -1.2$ with a vertical line. We also mark the intersection point $(x, \tilde{f}_X(x))$ to show the relative position of the target and proposal at that location. We add labels and a legend to clarify the plot.

```
%% 7. Visualize the target and proposal distributions

% Create new figure
figure
hold on
% Plot the target function (blue curve)
plot(x,target_function, ...
    'Color','blue', ...
    'DisplayName','Target function')
plot(x,proposal_uniform_PDF, ...
    'DisplayName','Proposal PDF')
% Plot the scaled proposal distribution (black curve)
plot(x,C*proposal_uniform_PDF, ...
    'Color','black', ...
    'DisplayName','Scaled proposal PDF')
plot(-4:-3,[max(target_function) max(target_function)],'--', ...
    'DisplayName','Max target function')
xline(-1.2,'--', ...
    'DisplayName','Reference x = -1.2')
f_Value = target_function((-1.2+4)/0.01); % Intersection point f(-1.2)
plot(-1.2,f_Value,'Marker','.', ...
    'DisplayName','Intersection f(-1.2)')
text(-5.1,max(target_function),'C \cdot p_X(-1.2)')
text(-1.3,-0.08,'-1.2')
title('Fig. 1. Target vs Proposal Distributions in Rejection Sampling')
xlabel('x')
```

```
90 legend('show');
91 hold off
```

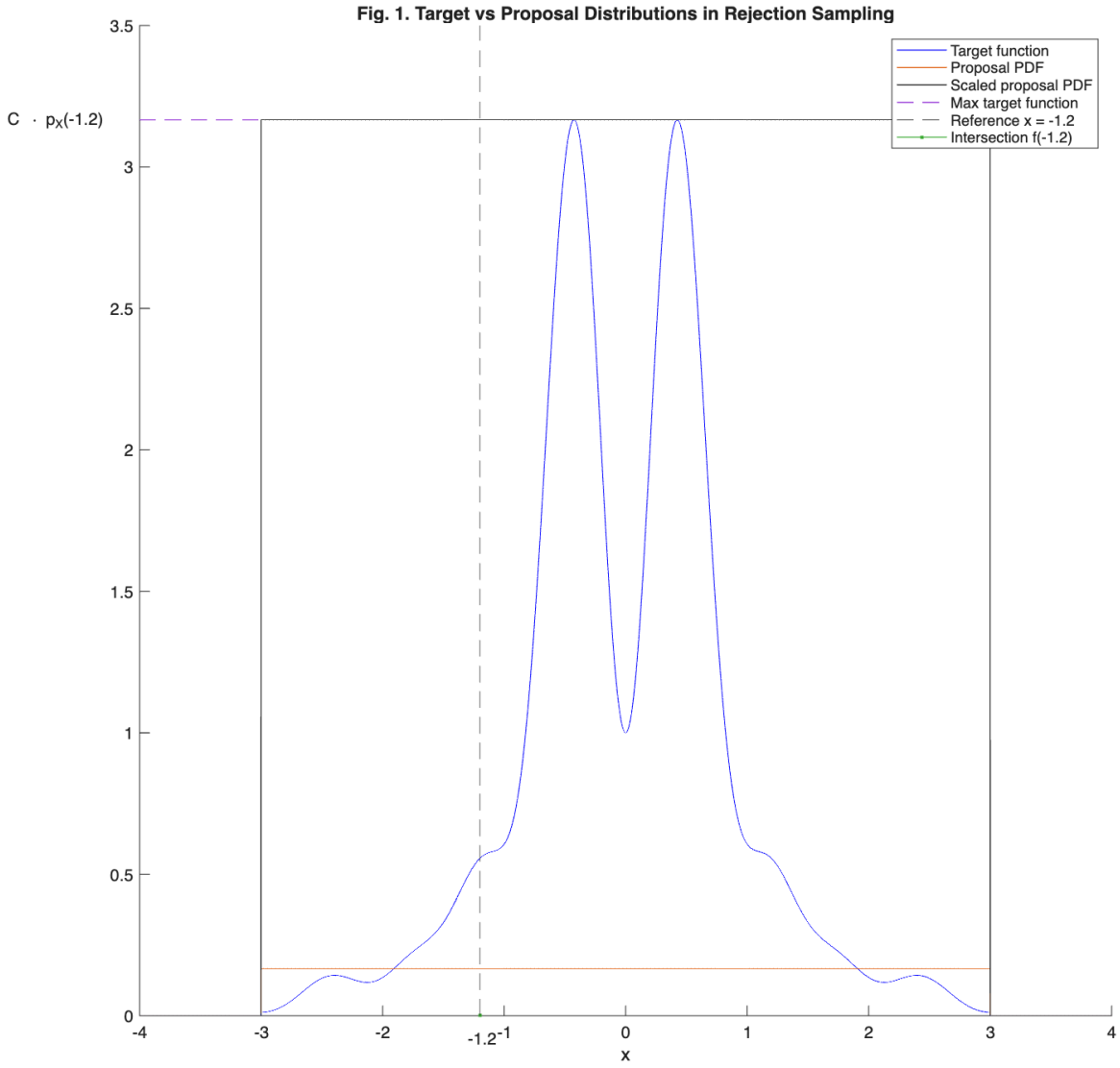
Fig. 1. Target vs Proposal Distributions in Rejection Sampling
Target function
Proposal PDF
Scaled proposal PDF
Max target function
Reference x = -1.2
Intersection f(-1.2)
C · pX(-1.2)
3.5
3
2.5
2
1.5
1
0.5
0
-4
-3
-2
-1.2
-1
0
1
2
3
4
x

Figure 1

8. Determine accepted and rejected samples

To perform the acceptance-rejection test, draw a random variable

$$Y \sim \text{Unif}\big(0,\, C \cdot p_X(-1.2)\big).$$

If $Y \leq \tilde{f}_X(-1.2)$, we accept $x = -1.2$; otherwise, we reject it. This procedure illustrates the acceptance-rejection criterion of rejection sampling. In neighborhoods around peaks of $\tilde{f}_X$, proposed values of x are more likely to be accepted, whereas in regions where $\tilde{f}_X$ is relatively small, they are more likely to be rejected.

Here we generalize this idea to many samples. First, we draw x-coordinates from the uniform proposal distribution on $[-3, 3]$. Second, we generate independent uniform variables $U \sim \text{Unif}(0, 1)$ to place points vertically under the envelope $C \cdot p_X$. Multiplying these factors yields y-coordinates for each proposal point. We then create a loop where we loop through all samples: if the y-coordinate lies below the target function at the corresponding x, we store the point as an accepted sample; otherwise, we store it as a rejected sample. In this way, the algorithm separates accepted and rejected draws according to the acceptance-rejection rule.

```
93  %% 8. Determine accepted and rejected samples
94
95  % 8.1. Generate x-coordinates from the uniform proposal on [-3,3]
96  proposal_samples = random('Uniform',-3,3,[N_samples 1]);
97
98  % 8.2. Generate U ~ Unif(0,1) for vertical placement
99  u_samples = random('Uniform',0,1,[N_samples 1]);
100
101 % 8.3. Compute corresponding y-coordinates scaled by envelope height
102 y_coordinates = u_samples.* ...
103     (C*pdf(proposal_uniform_PD,proposal_samples));
104
105 % 8.4. Combine x- and y-coordinates
106 proposal_points_scaled = [proposal_samples y_coordinates];
107
108 % 8.5. Preallocate matrices for accepted and rejected points
109 accepted_samples = NaN(N_samples,2);
110 rejected_samples = NaN(N_samples,2);
111
112 % 8.6. Classify each sample
113 for j = 1:N_samples
114     x_coordinate = proposal_points_scaled(j,1);
115     y_coordinate = proposal_points_scaled(j,2);
116     % Find index in x-values for target_function(x_coordinate)
117     find_index = (abs(min(x)) + x_coordinate) / 0.001;
118     index = int16(find_index) + 1; % Adjust for round-off error
119     if y_coordinate <= target_function(index)
120         % Accepted sample
121         accepted_samples(j,:) = [x_coordinate, y_coordinate];
122     else
123         % Rejected sample
124         rejected_samples(j,:) = [x_coordinate, y_coordinate];
```

```
    end
end
```

9. Visualize accepted and rejected samples

Figure 2 shows the outcome of the rejection sampling algorithm. Many points are rejected, plotted in black, because of the large difference in area between the proposal density p_X and the target function $\tilde{f}_X$. The accepted points are shown in blue. This illustrates that efficiency depends on how closely the proposal envelope matches the shape of the target.

```
%% 9. Visualize accepted and rejected samples

% Create new figure
figure
hold on
% Plot the target function (blue curve)
plot(x,target_function, ...
    'Color','blue', ...
    'DisplayName','Target function')
% Plot the scaled proposal distribution (black curve)
plot(x,C*proposal_uniform_PDF, ...
    'Color','black', ...
    'DisplayName','Scaled proposal')
% Plot accepted points (below target function, blue dots)
scatter(accepted_samples(:,1),accepted_samples(:,2),10, ...
    'MarkerEdgeColor','blue','MarkerFaceColor','blue', ...
    'DisplayName','Accepted samples')
% Plot rejected points (above target function, black dots)
scatter(rejected_samples(:,1),rejected_samples(:,2),10, ...
    'MarkerEdgeColor','black','MarkerFaceColor','black', ...
    'DisplayName','Rejected samples')
title('Fig. 2. Accepted vs Rejected Samples in Rejection Sampling')
xlabel('x')
legend('show')
hold off
```

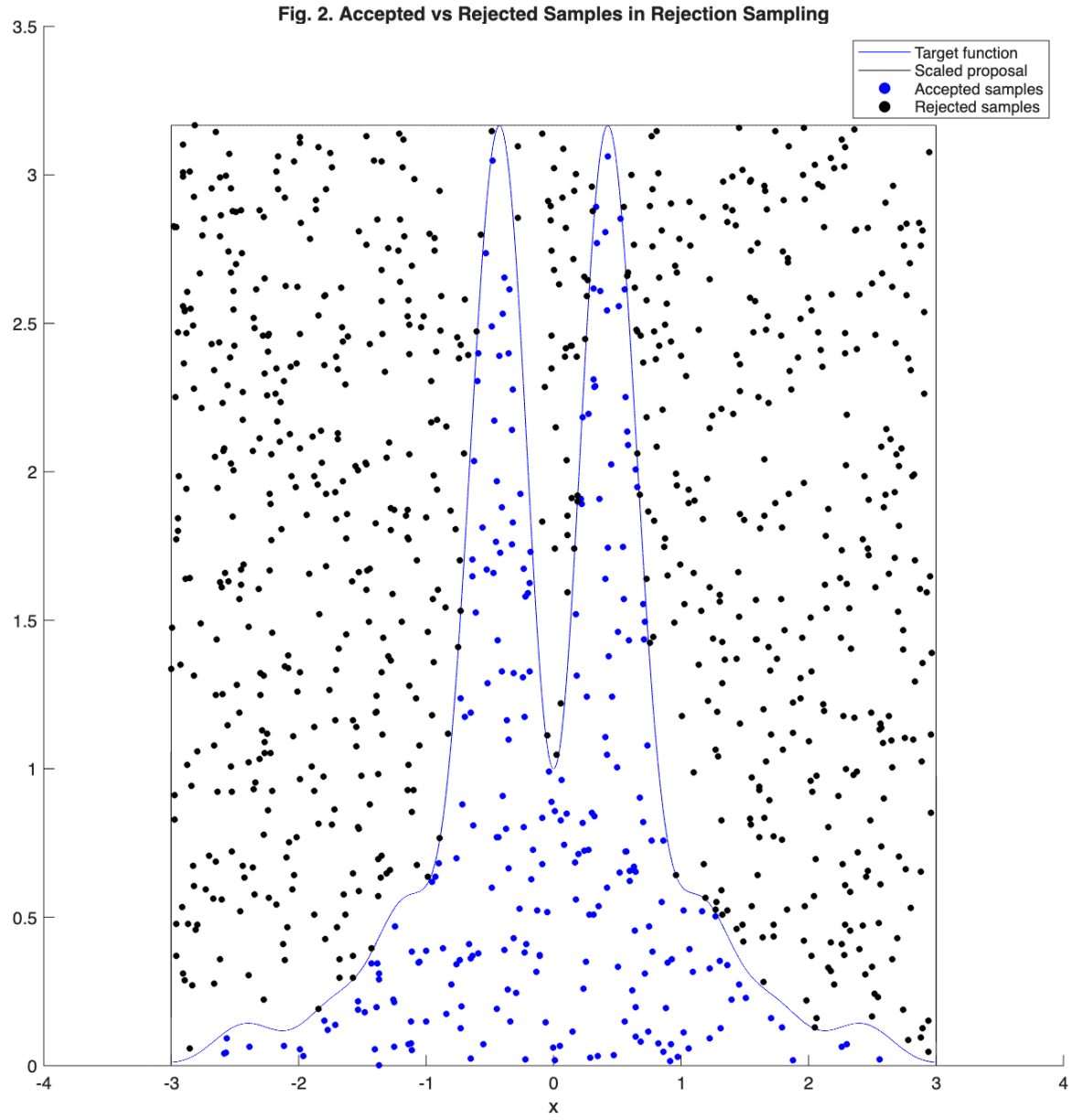
Fig. 2. Accepted vs Rejected Samples in Rejection Sampling
Target function
Scaled proposal
Accepted samples
Rejected samples
3.5
3
2.5
2
1.5
1
0.5
0
-4
-3
-2
-1
0
1
2
3
4
x

Figure 2

10. Construct a new proposal distribution

According to the general theory of rejection sampling, the method is valid for any proposal density p_X that satisfies two conditions. First, the support condition requires that the support of $\tilde{f}_X$ is contained in the support of p_X. Second, the envelope condition requires that there exists a constant $C \geq 1$ such that $\tilde{f}_X(x) \leq C \cdot p_X(x)$ for all x.

The uniform proposal used earlier satisfies these conditions but is inefficient: many points are rejected because the scaled uniform density does not resemble the shape of the target function $\tilde{f}_X$. To improve efficiency, we introduce a new proposal distribution that better matches the target. A natural choice is the standard normal distribution, $p_X \overset{d}{=} \mathcal{N}(0,1)$, since the target function itself contains a Gaussian component.

In the MATLAB code below, this new proposal is defined in three steps. First, the proposal distribution is specified as a standard normal with mean 0 and variance 1. Second, the corresponding PDF values are computed over the domain x. Third, an envelope constant C_N is chosen so that $C_N \cdot p_X(x)$ dominates $\tilde{f}_X(x)$ everywhere. By trial and error, one finds that $C_N = 9$ satisfies this condition, ensuring that the scaled normal proposal serves as a valid envelope for rejection sampling.

```
%% 10. Construct a new proposal distribution

% 10.1. Define the proposal distribution as a standard normal N(0,1)
proposal_standard_normal_PD = makedist('Normal','mu',0,'sigma',1);

% 10.2. Compute the proposal PDF values over the domain x
proposal_standard_normal_PDF = pdf(proposal_standard_normal_PD,x);

% 10.3. Choose envelope constant C_N so that C_N * proposal_PDF >=
% target_function
C_N = 9;
```

11. Visualize the target and new proposal distributions

Figure 3 illustrates the target function $\tilde{f}_X$ together with the scaled normal proposal distribution. The envelope constant $C_N = 9$ ensures that $C_N \cdot p_X(x)$ lies above $\tilde{f}_X(x)$ for all x in the support, thereby satisfying the envelope condition. Compared to the uniform proposal used earlier, the normal proposal provides a much tighter fit to the shape of the target function, which reduces the rejection rate and improves efficiency. In the code below we generate this visualization by plotting the target function in blue and the scaled normal proposal in black. The figure makes clear that the scaled normal proposal dominates the target function across its support while following its Gaussian shape more closely.

```
%% 11. Visualize the target and new proposal distributions

% Create new figure
figure
hold on
% Plot the target function (blue curve)
plot(x,target_function, ...
    'Color','blue', ...
```

```
    'DisplayName','Target function')
% Plot the scaled proposal distribution (black curve)
plot(x,C_N*proposal_standard_normal_PDF, ...
    'Color','black', ...
    'DisplayName','Scaled proposal')
title('Fig. 3. Target vs Scaled Normal Proposal in Rejection Sampling')
xlabel('x')
legend('show')
hold off
```

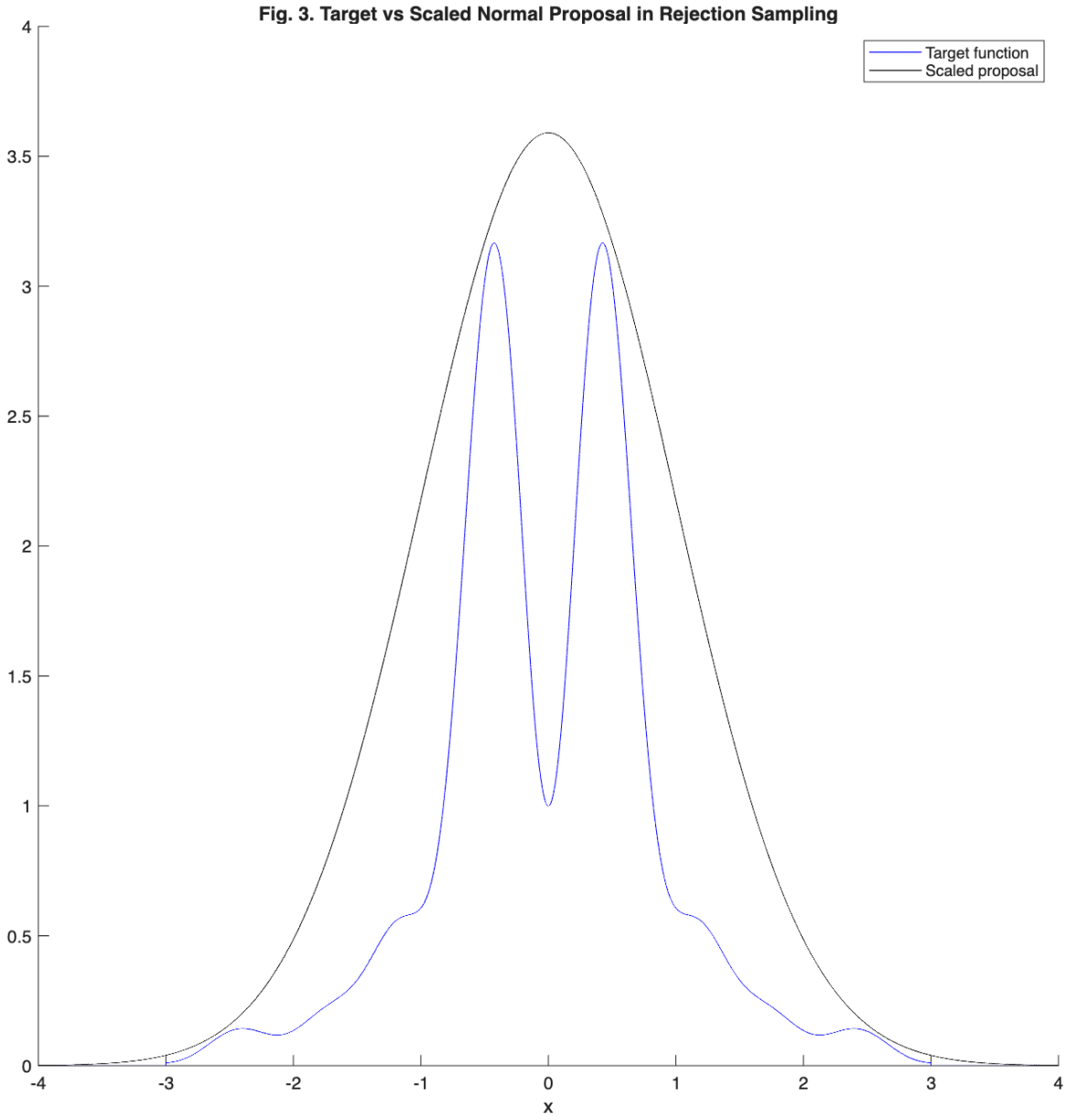
Fig. 3. Target vs Scaled Normal Proposal in Rejection Sampling
Target function
Scaled proposal
4
3.5
3
2.5
2
1.5
1
0.5
0
-4
-3
-2
-1
0
1
2
3
4
x

Figure 3

12. Determine accepted and rejected samples

To perform the acceptance-rejection test under the new proposal, we again draw a random variable

$$Y \sim \text{Unif}\big(0,\, C_N \cdot p_X(x)\big),$$

where p_X is the standard normal density and $C_N = 9$ is the envelope constant. For a candidate point x, we accept it whenever $Y \leq \tilde{f}_X(x)$; otherwise, we reject it. This procedure is identical in principle to the test described earlier, but the choice of proposal distribution changes the relative likelihood of acceptance across the domain.

Compared to Section 8, where the uniform proposal was used, the normal proposal aligns more closely with the Gaussian component of $\tilde{f}_X$. As a result, candidate points near the center of the distribution are more likely to be accepted, while fewer points are wasted in the tails. In the code below, we implement this by generating candidate x values from the normal proposal, drawing independent uniforms for the vertical placement, and classifying each point as accepted or rejected depending on whether it falls below the target function. The visualization shows accepted samples in blue and rejected samples in black, illustrating the improved efficiency of the normal proposal relative to the uniform case.

```
%% 12. Determine accepted and rejected samples

% 12.1. Generate x-coordinates (realizations from N(0,1))
proposal_samples = random('Normal',0,1,[N_samples 1]);

% 12.2. Compute corresponding y-coordinates
proposal_pdf_values = (1/sqrt(2*pi)) * ...
    exp(-0.5 * proposal_samples.^2);

% 12.3. Generate uniform scaling factors to spread points vertically
u_samples = random('Uniform',0,1,[N_samples 1]);

% 12.4. Define envelope constant (from Section 10)
c = C_N;

% 12.5. Scale y-coordinates by envelope constant and uniform factor
proposal_points_scaled = [proposal_samples ...
    c * u_samples .* proposal_pdf_values];

% 12.6. Preallocate matrix for accepted points (below target function)
accepted_samples = NaN(N_samples,2);

% 12.7. Preallocate matrix for rejected points (above target function)
rejected_samples = NaN(N_samples,2);

% 12.8. Loop through all samples and classify each point
for k = 1:N_samples
    x_coordinate = proposal_points_scaled(k,1);
    y_coordinate = proposal_points_scaled(k,2);
    find_index = (abs(min(x)) + x_coordinate) / 0.001;
    index = int16(find_index) + 1; % Adjust for round-off error
```

```
    if y_coordinate <= target_function(index)
        accepted_samples(k,:) = [x_coordinate, y_coordinate];
    else
        rejected_samples(k,:) = [x_coordinate, y_coordinate];
    end
end
```

13. Visualize accepted and rejected samples

The comparison between the uniform and normal proposals highlights the importance of choosing an efficient envelope. With the uniform proposal, many candidate points were rejected because the envelope did not resemble the target function $\tilde{f}_X$. By contrast, the normal proposal provided a tighter fit, reducing the rejection rate and improving efficiency. This demonstrates that the closer the proposal density p_X matches the shape of the target, the smaller the required envelope constant C and the higher the acceptance probability.

```
%% 13. Visualize accepted and rejected samples

% 13.1. Create a new figure window
figure
hold on
% Plot the target function (blue curve)
plot(x,target_function, ...
    'Color','blue', ...
    'DisplayName','Target function')
% Plot the scaled standard normal proposal distribution (black curve)
plot(x,C_N*proposal_standard_normal_PDF, ...
    'Color','black', ...
    'DisplayName','Scaled proposal')
% Plot accepted samples (blue dots below target function)
scatter(accepted_samples(:,1),accepted_samples(:,2),10, ...
    'MarkerEdgeColor','blue','MarkerFaceColor','blue', ...
    'DisplayName','Accepted samples')
% Plot rejected samples (black dots above target function)
scatter(rejected_samples(:,1),rejected_samples(:,2),10, ...
    'MarkerEdgeColor','black','MarkerFaceColor','black', ...
    'DisplayName','Rejected samples')
title('Fig. 4. Accepted vs Rejected Samples with Normal Proposal')
xlabel('x')
legend('show')
hold off
```

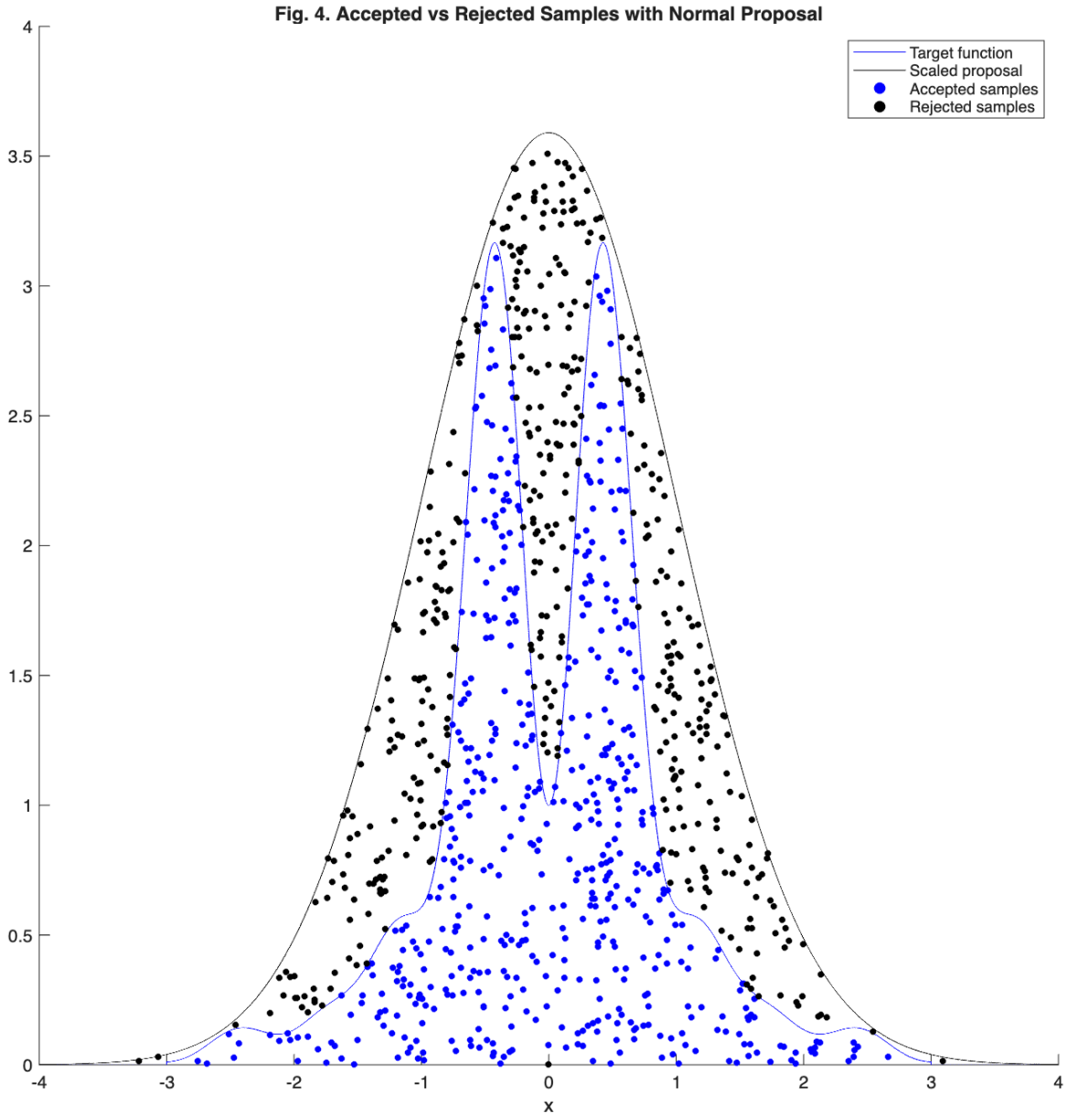
Fig. 4. Accepted vs Rejected Samples with Normal Proposal
Target function
Scaled proposal
Accepted samples
Rejected samples
4
3.5
3
2.5
2
1.5
1
0.5
0
-4
-3
-2
-1
0
1
2
3
4
x

14. Final notes