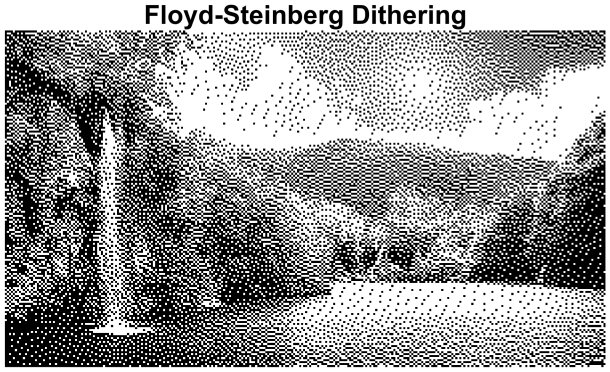
Question 2: Floyd-Steinberg and Jarvis-Judice-Ninke Dithering

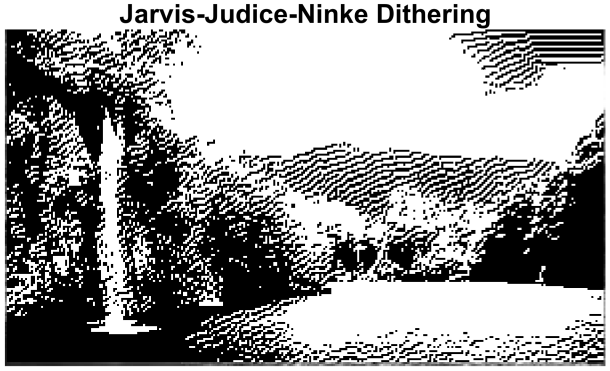
```
% Load the image
image = imread('Nature.jpeg');
gray_image = rgb2gray(image); % Convert to grayscale

% Floyd-Steinberg Dithering
floyd_dithered = dither(gray_image);
figure, imshow(floyd_dithered), title('Floyd-Steinberg Dithering');
```




```
jjn_filter = [0 0 0; 0 0 7; 3 5 1] / 48; % Filter matrix
jjn_dithered = double(gray_image);
for i = 1:size(jjn_dithered, 1) - 2
    for j = 2:size(jjn_dithered, 2) - 1
        old_pixel = jjn_dithered(i, j);
        new_pixel = round(old_pixel / 255) * 255;
        jjn_dithered(i, j) = new_pixel;
        error = old_pixel - new_pixel;
        jjn_dithered(i:i+2, j-1:j+1) = jjn_dithered(i:i+2, j-1:j+1) + error *
jjn_filter;
    end
end
figure, imshow(uint8(jjn_dithered)), title('Jarvis-Judice-Ninke Dithering');
```

Jarvis-Judice-Ninke Dithering



Explanation of Dithering Algorithms:1. Floyd-Steinberg Dithering:

- **Concept**: Floyd-Steinberg is an error diffusion dithering algorithm. It converts a grayscale image into a binary (black-and-white) image by distributing the quantization error of a pixel to neighboring pixels.
- **Method**:
- For each pixel, the algorithm rounds the grayscale value to the nearest black or white value.
- The quantization error is spread to nearby pixels (typically right, bottom, and diagonals) based on predefined weights.

2. Jarvis-Judice-Ninke (JJN) Dithering:

- **Concept**: JJN is a more complex dithering algorithm that also spreads quantization error to neighboring pixels, but over a **larger area** compared to Floyd-Steinberg.
- **Method**:
- The error is distributed to pixels up to two rows below and two columns to the right, using a larger kernel with more weights.
- This tends to produce smoother gradients at the cost of computational complexity.

Comparison:

1. **Floyd-Steinberg Dithering**:

- **Image Appearance**:
- The image looks more grainy, with a **noisy texture**.
- It introduces many small dots to approximate grayscale values, leading to a high level of detail but with visible noise.
- The edges, like those in the trees and waterfall, are somewhat sharp but with noticeable noise around them.
- **Strengths**:
- Good for small, detailed textures.
- Fast and computationally efficient.

- **Limitations**:
- Can introduce a noisy appearance, as seen in your output.

1. **Jarvis-Judice-Ninke Dithering**:

- **Image Appearance**:
- The result from JJN dithering is **smoother** and **less noisy**.
- You can see larger continuous patterns, which helps with better gradient approximation (e.g., sky and water).
- The waterfall edges look more polished compared to the Floyd-Steinberg output, with fewer small dots scattered around.
- **Strengths**:
- Smooth gradients and better quality in larger areas.
- Reduces visual noise compared to Floyd-Steinberg.
- **Limitations**:
- More computationally expensive.
- May introduce some visible banding or pattern artifacts in some areas, as seen in the top right of the JJN output.

Conclusion:

- **Floyd-Steinberg** dithering gives better results for **high-detail areas** but introduces more noise.
- **JJN dithering** provides a smoother result and works well with **gradients** or areas with fewer details.
- From image, **JJN** is better for scenes with continuous changes (like landscapes) since it maintains a smoother appearance, whereas **Floyd-Steinberg** is useful for detailed, intricate patterns but with a tradeoff in noise.

This comparison helps explain the **visual differences between the two methods**. Let me know if further clarification is needed!