APMA 4301: Project Report

Aditya Jabade

1 Problem Statement & Background

Image denoising is a fundamental task in signal and image processing, where the goal is to suppress noise while preserving essential structural features. The heat equation, a classic example of a parabolic partial differential equation, offers a physically meaningful approach to smoothing. It models the smoothing of pixel intensities as a diffusion process, making it a natural fit for image denoising. By solving the heat equation numerically, we can simulate the gradual removal of high-frequency noise.

The objective of this work is to **investigate the application of the two-dimensional heat equation for image denoising**. Specifically, the aim is to model the evolution of pixel intensities under (isotropic) diffusion and evaluate how different numerical schemes — including the Forward Euler, Backward Euler, Crank–Nicolson and the Method-of-Lines (MoL) — affect the quality and stability of the denoised output. The work focuses on applying these schemes to synthetically generated binary images corrupted with Gaussian noise. Performance is evaluated primarily through visual inspection to assess denoising quality and scheme behavior.

2 Mathematical Formulation

The two dimensional heat equation is:

$$\frac{\partial u}{\partial t} = \alpha \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right] \quad ; \quad u = u(x, y, t) \tag{1}$$

The noisy image evolving with the above dynamics can be modeled as follows:

Mathematical variable	**Physical counterpart**
$u(x, y, t)_{0 \le x \le w, 0 \le y \le h}$	Image of size $w \times h$ at time t
$u(x_i, y_j, t)$	Intensity of pixel (x_i, y_j) at time t
α	Diffusion coefficient (Controls degree of smoothing)
$u(x, y, 0)_{0 \le x \le w, 0 \le y \le h}$	Initial Noisy image

The initial condition is thus the noisy image. Consider that the image is of dimensions $w \times h$. The boundary conditions are essentially the intensity values of the boundary of the outermost 1-pixel area. This work focuses exclusively on **Neumann boundary conditions**, which enforce a zero-gradient constraint at the image edges. This effectively mirrors adjacent interior pixels, ensuring smooth continuation at the boundaries. Dirichlet conditions—where the boundary values are fixed—are less appropriate in this context, as there is typically no justified reason to force the outermost pixels to a predefined value unless the image includes a known external frame or margin.

3 Numerical Methods

The following numerical methods are used to discretize Equation 1 in time. In all cases, the **spatial derivatives are approximated using centered difference schemes**, and the boundary conditions are taken to be **homogeneous Neumann**: $\frac{\partial u}{\partial n}\big|_{\partial\Omega} = 0$, where $\partial\Omega$ denotes the boundary of the image domain and $\frac{\partial u}{\partial n}$ is the derivative in the direction normal to the boundary. Essentially for our $w \times h$ image this means:

- **Top edge:** $u(0, j) = u(1, j) \;\; ; 0 \le j \le h$
- **Bottom edge:** $u(w-1, j) = u(w-2, j) \;\; ; 0 \le j \le h$
- **Left edge:** $u(i, 0) = u(i, 1) \;\; ; 0 \le i \le w$

- **Right edge:** $u(i, h-1) = u(i, h-2) \;\; ; 0 \leq i \leq w$

1. **Forward Euler:**

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \alpha \Delta t \left[\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{h^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{k^2} \right]$$

   This explicit scheme is solved iteratively using a uniform grid spacing $h = k$.

2. **Backward Euler:**

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \alpha \Delta t \left[\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{h^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{k^2} \right]$$

$$\therefore u_{i,j}^{n+1} - \alpha \Delta t \left[\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{h^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{k^2} \right] = u_{i,j}^{n}$$

   This is an implicit system where each unknown $u_{i,j}^{n+1}$ depends on its four neighboring points at the next time step. To compute the full solution at time $n+1$, we must solve a linear system: $\mathbf{A} \cdot \mathbf{u}^{n+1} = \mathbf{u}^{n}$, where $\mathbf{u}^{n+1}$ and $\mathbf{u}^{n}$ are the flattened (1D) representations of the 2D grid of values at time steps $n+1$ and n, respectively. The matrix $A \in \mathbb{R}^{(w \times h) \times (w \times h)}$ captures the 5-point stencil structure. For a uniform grid spacing $h = k$, the matrix A is sparse and symmetric. Each row of A corresponds to a grid point and includes:

   - A diagonal entry $1 + 4\lambda$
   - Off-diagonal entries $-\lambda$ for each of the four valid neighbors,
     where $\lambda = \frac{\alpha \Delta t}{h^2}$

$$A = \begin{bmatrix} 1+4\lambda & -\lambda & 0 & \cdots & -\lambda & 0 & \cdots \\ -\lambda & 1+4\lambda & -\lambda & \cdots & 0 & -\lambda & \cdots \\ 0 & -\lambda & 1+4\lambda & \cdots & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \\ -\lambda & 0 & 0 & \cdots & 1+4\lambda & -\lambda & \cdots \\ 0 & -\lambda & 0 & \cdots & -\lambda & 1+4\lambda & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \end{bmatrix}$$

3. **Crank-Nicholson:** In this method we combine the Forward and Backward Euler schemes in a fixed proportion ($\theta = 0.5$):

$$\text{Crank-Nicholson} = \theta \cdot \text{Forward Euler} + (1-\theta) \cdot \text{Backward Euler}$$

$$\begin{aligned} \therefore u_{i,j}^{n+1} \; &- \frac{\alpha \Delta t}{2} \left[\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{h^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{k^2} \right] \\ &= u_{i,j}^{n} + \frac{\alpha \Delta t}{2} \left[\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{h^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{k^2} \right] \end{aligned}$$

   This requires solving a linear system at each time step, just like the Backward Euler method. The construction follows from flattening the image matrix u and using the 5-point stencil as before. Thus we have the follwoing :

$$\left(I - \frac{\lambda}{2} L \right) \mathbf{u}^{n+1} = \left(I + \frac{\lambda}{2} L \right) \mathbf{u}^{n}$$

   where L represents the discrete 2D Laplacian matrix from the 5-point stencil and $\lambda = \frac{\alpha \Delta t}{h^2}$. Essentially:

$$\mathbf{A} \cdot \mathbf{u}^{n+1} = \mathbf{B} \cdot \mathbf{u}^{n} \quad ; \mathbf{A}, \mathbf{B} \in R^{(w \times h) \times (w \times h)} \text{ and } \mathbf{u}^{n+1}, \mathbf{u}^{n} \in R^{w \times h}$$

$$A = \begin{bmatrix} 1-2\lambda & \frac{\lambda}{2} & 0 & \cdots & \cdots & \frac{\lambda}{2} & \cdots \\ \frac{\lambda}{2} & 1-2\lambda & \frac{\lambda}{2} & \cdots & \cdots & 0 & \cdots \\ 0 & \frac{\lambda}{2} & 1-2\lambda & \cdots & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \\ \frac{\lambda}{2} & 0 & 0 & \cdots & 1-2\lambda & \frac{\lambda}{2} & \cdots \\ 0 & \frac{\lambda}{2} & 0 & \cdots & \frac{\lambda}{2} & 1-2\lambda & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$B = \begin{bmatrix} 1+2\lambda & -\frac{\lambda}{2} & 0 & \cdots & \cdots & -\frac{\lambda}{2} & \cdots \\ -\frac{\lambda}{2} & 1+2\lambda & -\frac{\lambda}{2} & \cdots & \cdots & 0 & \cdots \\ 0 & -\frac{\lambda}{2} & 1+2\lambda & \cdots & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \\ -\frac{\lambda}{2} & 0 & 0 & \cdots & 1+2\lambda & -\frac{\lambda}{2} & \cdots \\ 0 & -\frac{\lambda}{2} & 0 & \cdots & -\frac{\lambda}{2} & 1+2\lambda & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \end{bmatrix}$$

4. **Method-of-Lines**: In this method we first semi-discretize the system in space to get the following equation: $\frac{du}{dt}(t) = \alpha L u(t)$ and then fully discretize in time using a chosen scheme (described in the next section). Here $L \in R^{(w\times h)\times(w\times h)}$ is constructed from the 5-point stencil assuming uniform grid spacing $h = k$.

$$L = \begin{bmatrix} 4 & -1 & 0 & \cdots & -1 & 0 & \cdots \\ -1 & 4 & -1 & \cdots & 0 & -1 & \cdots \\ 0 & -1 & 4 & -1 & \cdots & -1 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ -1 & 0 & \cdots & -1 & 4 & -1 & \cdots \\ 0 & -1 & 0 & \cdots & -1 & 4 & \cdots \\ \vdots & \vdots & \cdots & \ddots & \vdots & \vdots & \ddots \end{bmatrix}$$

4 Implementation

All numerical methods were implemented in Python using NumPy and SciPy. The image is treated as a 2D array of binary intensity values, and the goal is to evolve this image over time using the 2D heat equation. In all methods, homogeneous Neumann boundary conditions are imposed by padding the input image with a 1-pixel mirrored border using `np.pad(..., mode='edge')` and cropping it after computation.

- **Forward Euler**: An explicit scheme is implemented by updating only interior pixels using a 5-point stencil. The update rule is applied iteratively for a fixed number of time steps.
- **Backward Euler**: This implicit scheme requires solving a (sparse) linear system at each time step. This matrix system $Au^{n+1} = u^n$ is solved iteratively using SciPy's sparse direct solver `spsolve`.
- **Crank–Nicolson**: This combines the Forward and Backward Euler schemes with equal weights ($\theta = 0.5$). It solves the system $Au^{n+1} = Bu^n$ at each time step, where both A and B are constructed from the discrete Laplacian matrix.
- **Method of Lines (MoL)**: This approach first discretizes space but leaves time continuous. The flattened Laplacian operator is used to form an ODE system $\frac{du}{dt} = \alpha Lu$, which is then solved using `scipy.integrate.solve_iv` with the Runge–Kutta method (`RK45`).

Across all schemes, denoising performance is evaluated primarily through visual inspection, using surface plots and intensity maps to assess smoothness, edge behavior, and convergence. Parametric sensitivity studies are conducted to analyze stability and scheme behavior, while quantitative evaluation is left for future work.

5 Results

Fig. 1 shows the synthetically generated input image (256×256) alongside the noisy version, created by adding white Gaussian noise with a specified coefficient. This served as the test case for all subsequent denoising experiments.

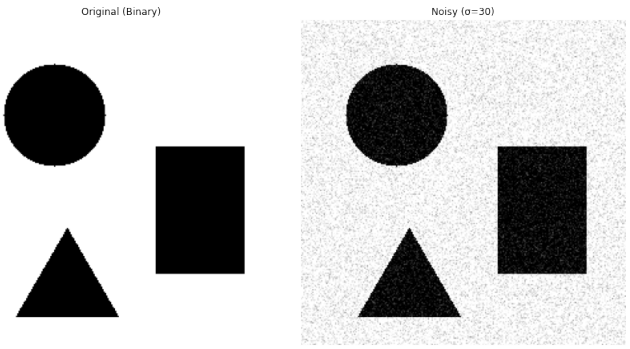

Figure 1: Inputs

Fig. 2 shows the output obtained by applying the Forward Euler scheme with fixed parameters.

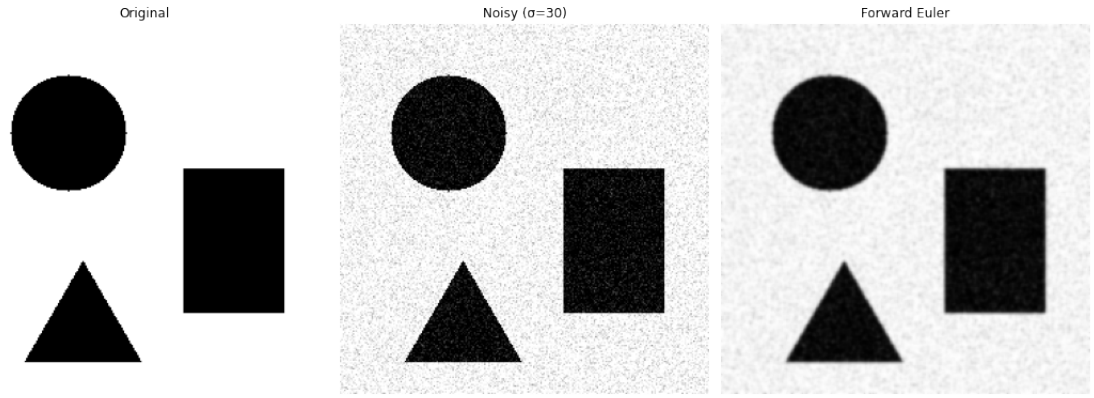

Figure 2: Forward Euler: Output

Fig. 3 displays the surface plot of the Forward Euler output, capturing the smoothened intensity landscape.

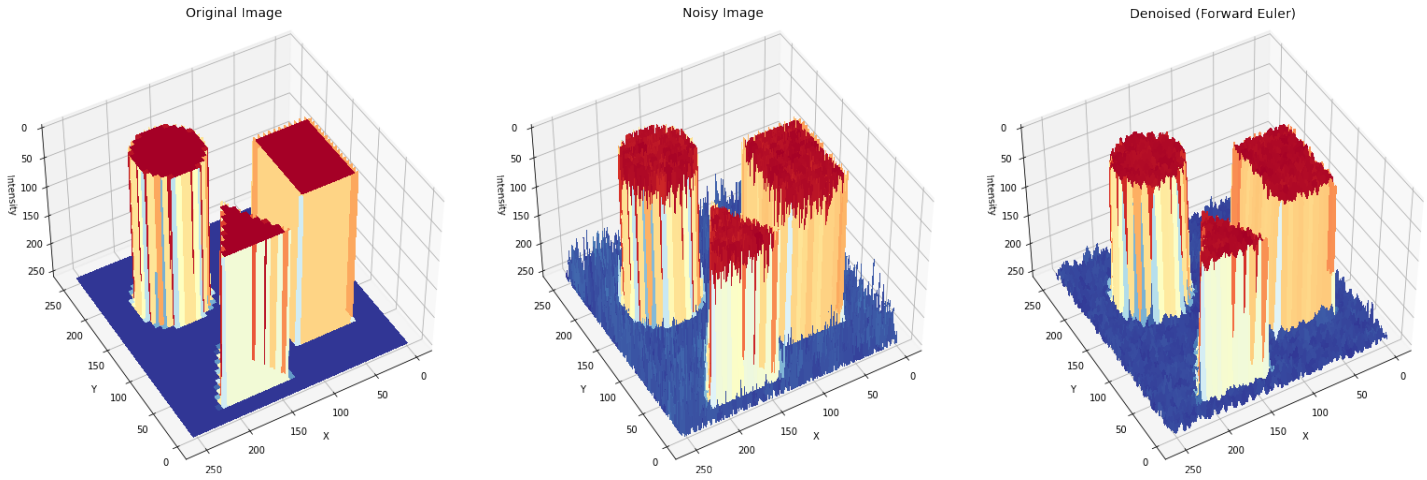

Figure 3: Forward Euler: Output Surface Plot

Fig. 4 shows how the Forward Euler results change as the time-step `dt` is varied.

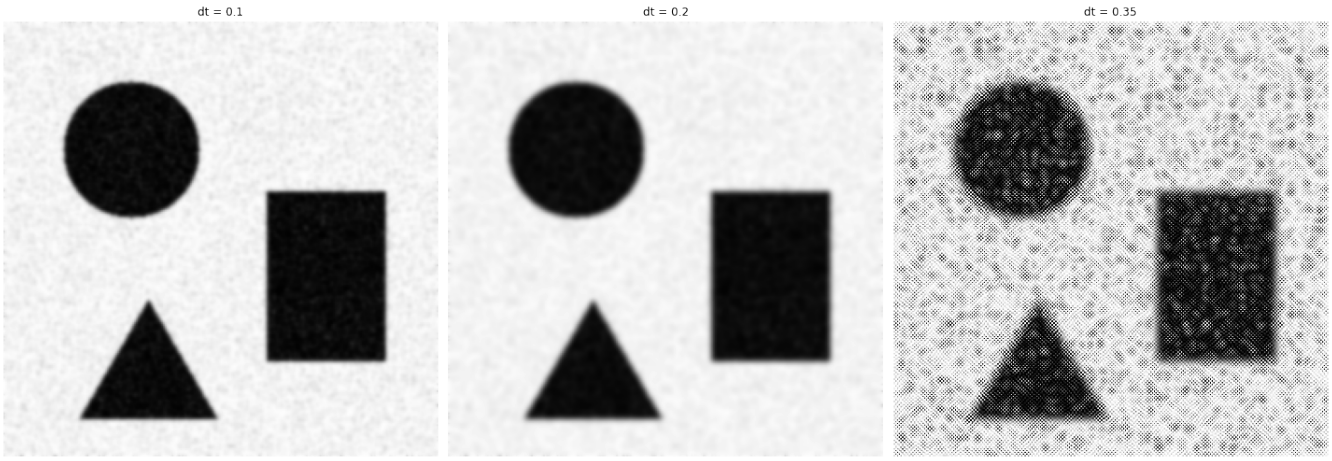

Figure 4: Forward Euler: Varying time-step

Fig. 5 illustrates the outputs obtained using different numbers of iterations in the Forward Euler method.

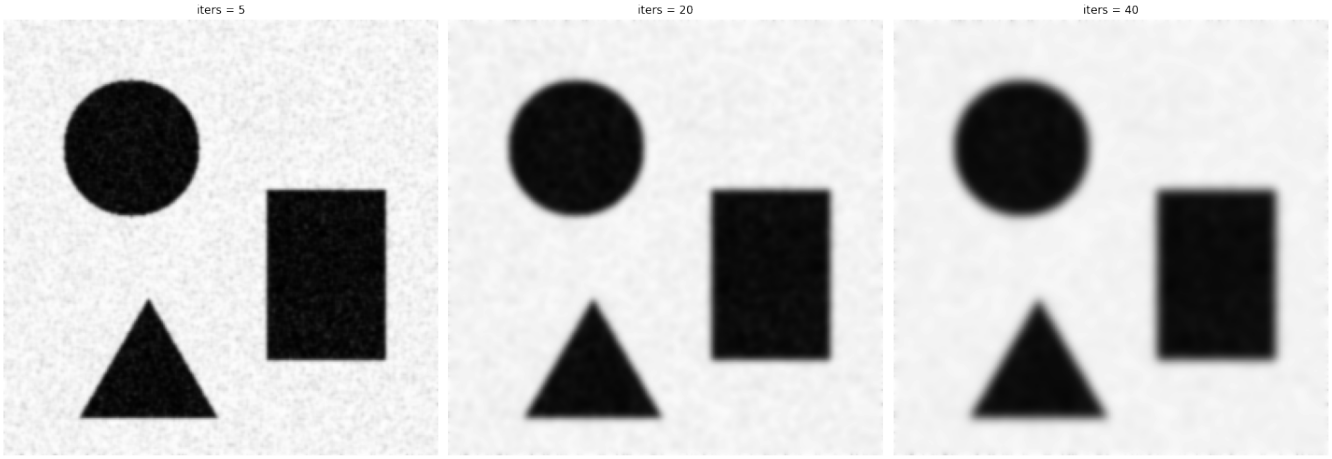

Figure 5: Forward Euler: Varying no. of iterations

Fig. 6 demonstrates how the output evolves with increasing values of the diffusion coefficient `alpha`.

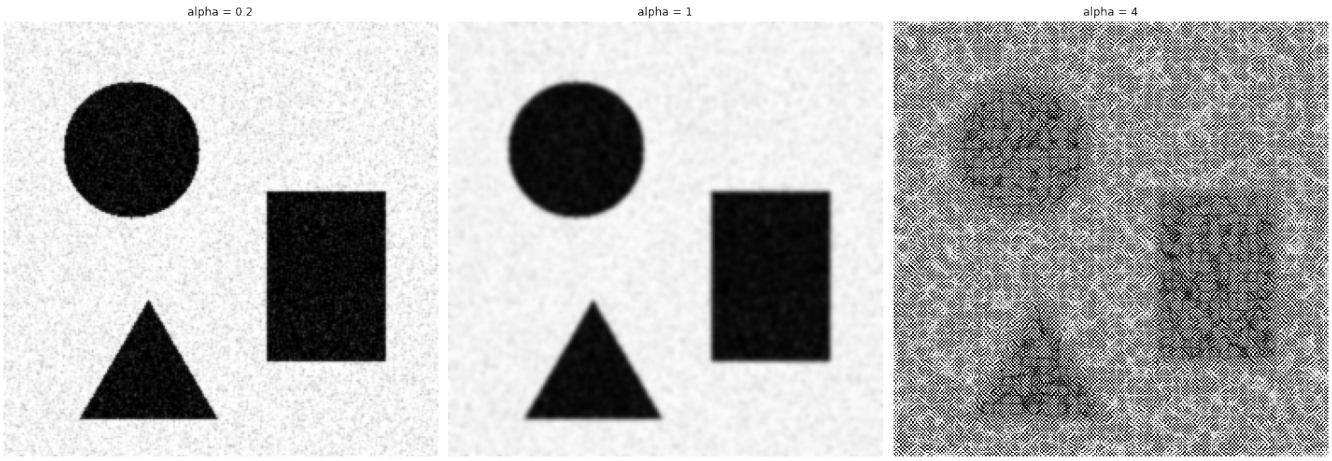

Figure 6: Forward Euler: Varying alpha

Fig. 7 presents the denoised image generated by the Backward Euler method using the same initial conditions.

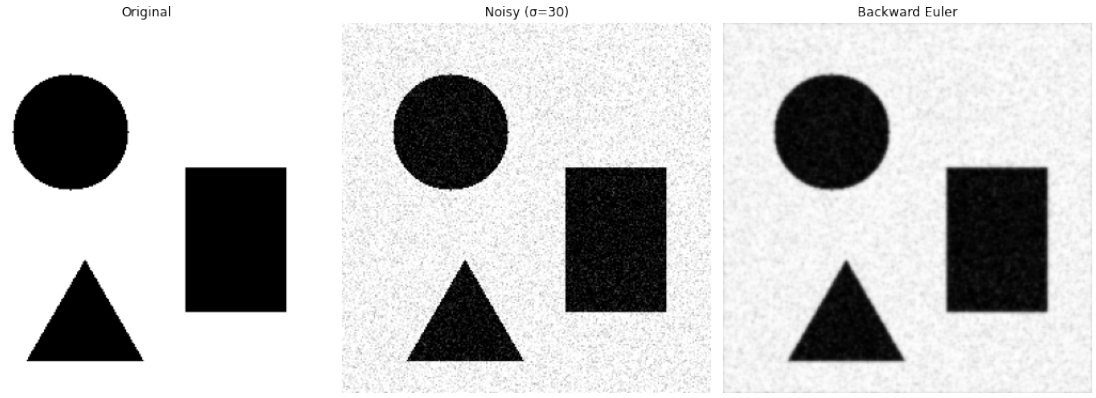

Figure 7: Backward Euler: Output

Fig. 8 shows the surface representation of the Backward Euler result, indicating the level of smoothness achieved.

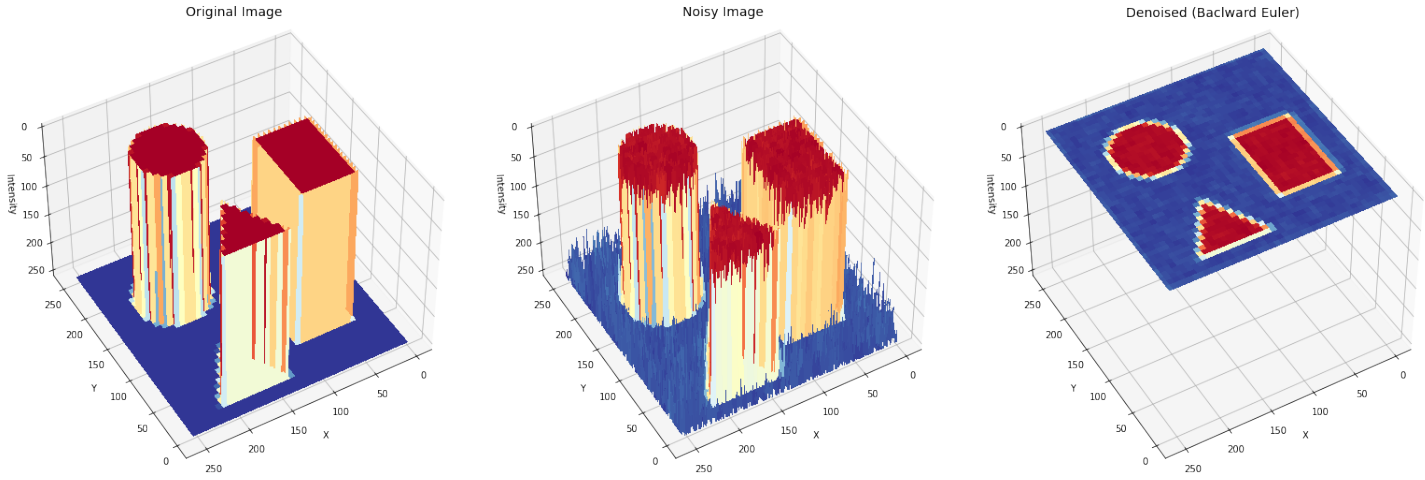

Figure 8: Backward Euler: Output Surface Plot

Fig. 9 illustrates how the Backward Euler output changes with different values of the time-step `dt`.

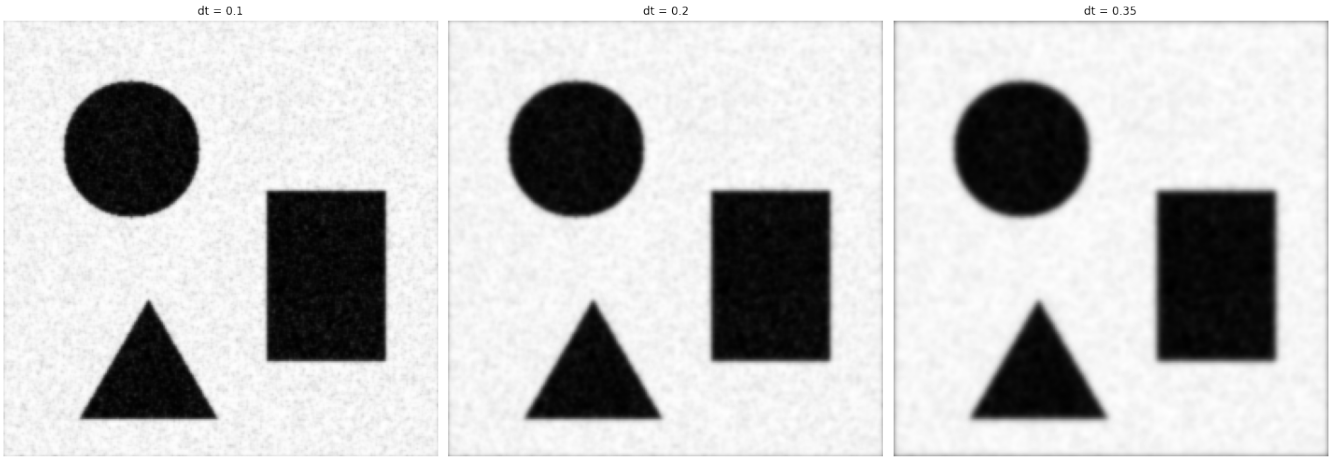

Figure 9: Backward Euler: Varying time-step

Fig. 10 shows the effect of varying the number of iterations on the Backward Euler denoising result.

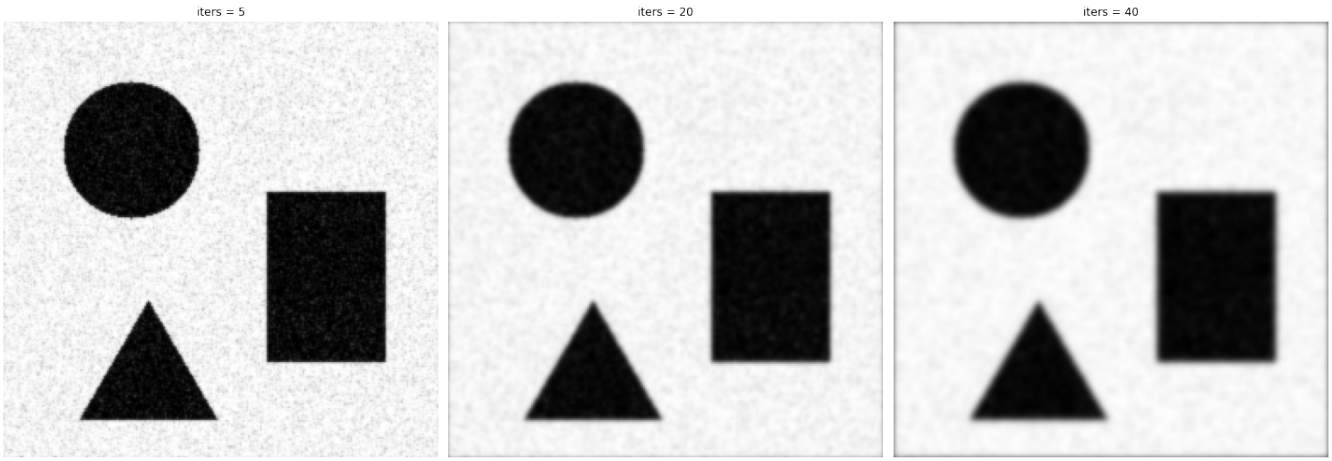

Figure 10: Backward Euler: Varying no. of iterations

Fig. 11 demonstrates the effect of increasing the diffusion parameter `alpha` in the Backward Euler method.

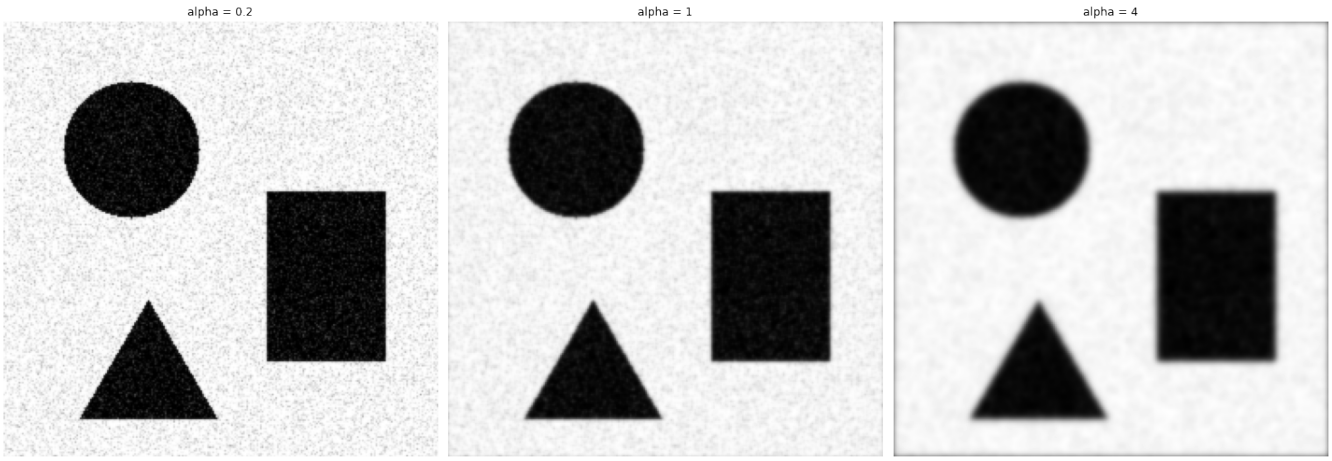

Figure 11: Backward Euler: Varying alpha

Fig. 12 presents the result of applying the Crank–Nicolson scheme to the padded noisy input image.

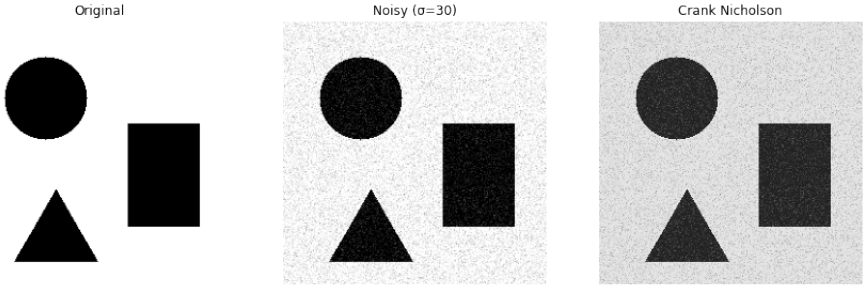

Figure 12: Crank Nicholson: Output

Fig. 13 shows the corresponding surface plot of the Crank–Nicolson output, visualizing intensity variation.

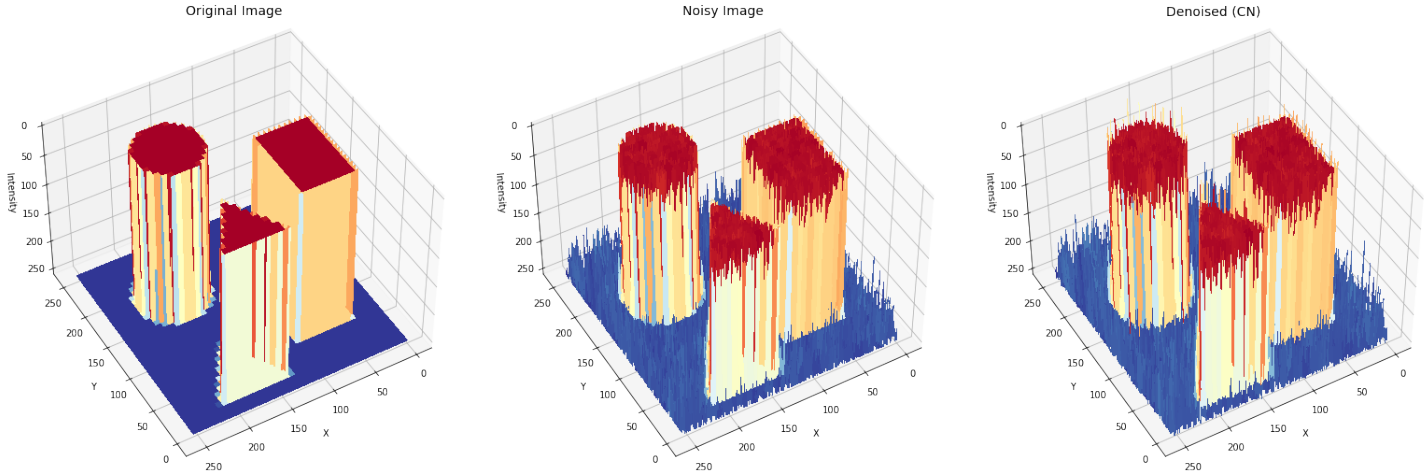

Figure 13: Crank Nicholson: Output Surface Plot

Fig. 14 shows the outputs of the Crank–Nicolson with different values of the time-step `dt`.

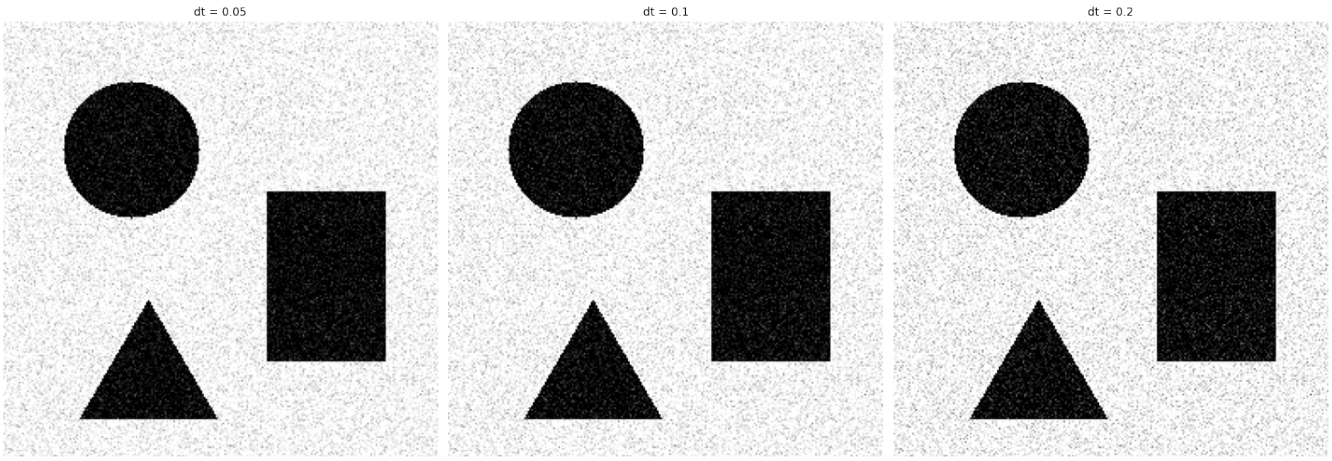

Figure 14: Crank Nicholson: Varying time-step

Fig. 15 illustrates the effect of varying the number of iterations in the Crank–Nicolson scheme.

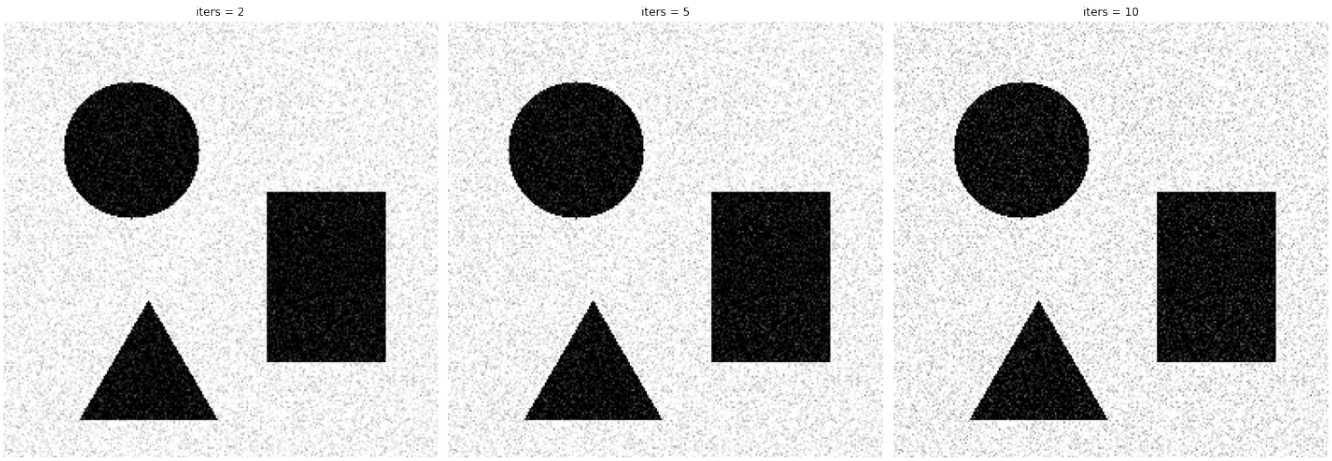

Figure 15: Crank Nicholson: Varying no. of iterations

Fig. 16 demonstrates how changing `alpha` influences the Crank–Nicolson behavior.

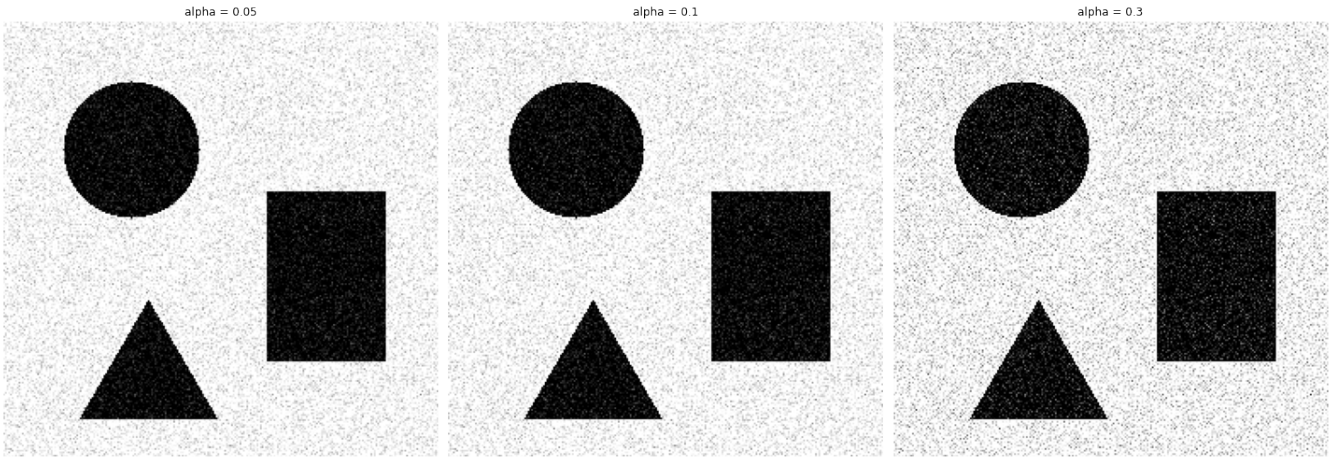

Figure 16: Crank Nicholson: Varying alpha

Fig. 17 shows the denoised result from the Method of Lines (MoL) approach using Runge–Kutta time integration.

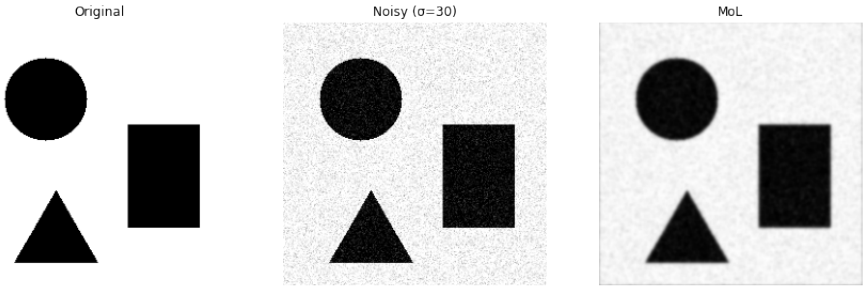

Figure 17: Method of Lines: Output

Fig. 18 presents a surface plot of the Method of Lines (MoL), capturing smoothed spatial variations.

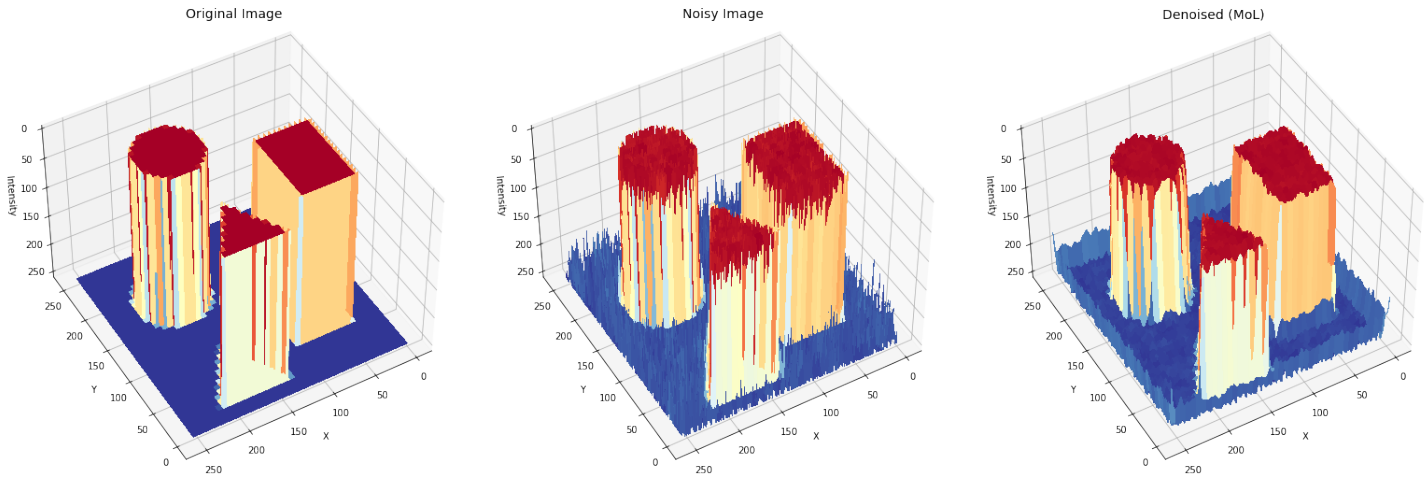

Figure 18: Method of Lines: Output Surface Plot

Fig. 19 illustrates the final outputs for different simulation durations `t_span` in the Method of Lines (MoL) method.

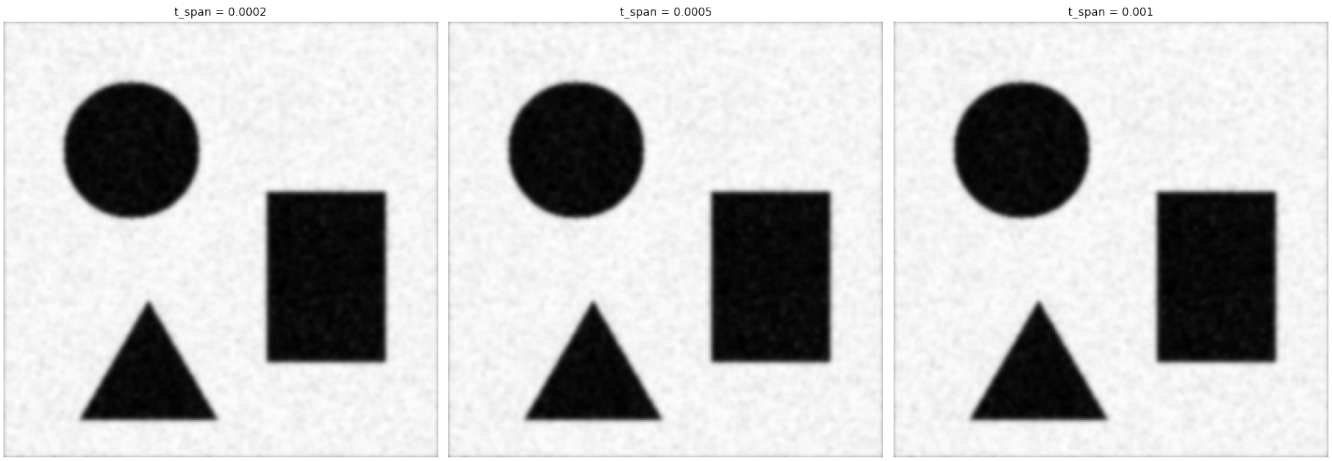

Figure 19: Method of Lines: Varying time-span

Fig. 20 shows outputs for varying numbers of evaluation points `t_eval` within the fixed time span.

Figure 20: Method of Lines: Varying time resolution

Fig. 21 demonstrates how the final denoised output changes with different values of `alpha` using the Method of Lines (MoL).

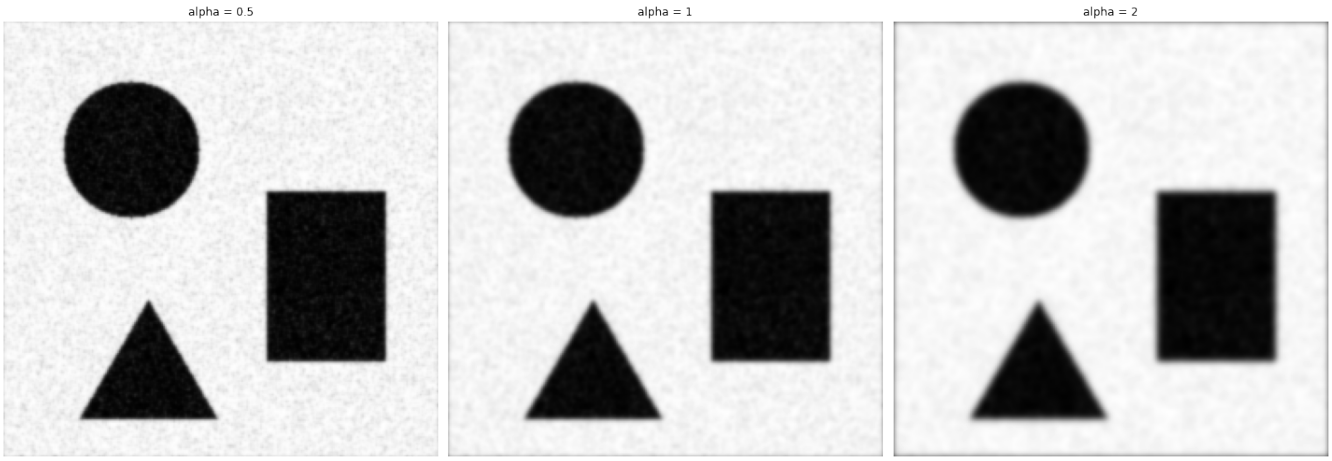

Figure 21: Method of Lines: Varying alpha

6 Discussion

The results illustrate how different numerical schemes influence the denoising behavior of the 2D heat equation across various parameter settings. Each method—Forward Euler, Backward Euler, Crank–Nicolson, and Method of Lines (MoL)—exhibited distinct characteristics in terms of smoothness, stability, and sensitivity.

The **Forward Euler** scheme, being explicit, required small time steps to maintain numerical stability due to the **CFL condition** $\alpha \frac{\Delta t}{\Delta x^2} \leq \frac{1}{4}$. As seen in Figs. 4 and 6, increasing `dt` or α beyond this threshold resulted in noticeable degradation or instability. Similarly, Fig. 5 and Fig. 6 show that increasing the number of iterations or the diffusion coefficient `alpha` respectively led to progressively stronger denoising, but at the cost of potential over-smoothing.

In contrast, the **Backward Euler** method is unconditionally stable and not restricted by a CFL condition, allowing larger time steps without instability (Figs. 9 and 11). However, as shown in Fig. 10 and Fig. 11, the scheme remains sensitive to iteration count and the diffusion coefficient, with even moderate increases producing significantly smoother outputs. The results suggest that while numerical stability is guaranteed, careful parameter tuning is essential to preserve important image details and avoid edge artifacts.

The **Crank–Nicolson** method with $\theta = 0.5$, proved counterproductive as seen in Fig. 12. The corresponding surface plot in Fig. 13 provides a clearer visualization, where the noise appears to be amplified rather than suppressed. Further sensitivity plots (Fig. 14–16) reinforce this observation, showing noise amplification across different parameter settings.

The **Method of Lines (MoL)** provided a continuous-in-time formulation, which was numerically integrated using the Runge–Kutta method. Figs. 19–21 illustrate how varying the total simulation time, number of evaluation points, and diffusion coefficient influenced the denoising outcome. MoL offered a fine control over temporal evolution and produced optimal smoothing results. However, it also exhibited significant edge darkening effects, particularly at higher time resolutions and larger `alpha` values.

Overall, we can say that while all four schemes are capable of evolving the heat equation for image denoising, their effectiveness varies significantly with tuning. Forward Euler is simple, efficient, and fast, but is restricted by a CFL condition and prone to instability or over-smoothing. Backward Euler is known to be unconditionally stable and demonstrated robust behavior. However, it took higher computational cost compared to Forward Euler and remained sensitive to parameter values, which could lead to excessive smoothing or edge-related artifacts. Crank–Nicolson exhibited unexpected amplification of noise, failed to suppress high-frequency components across different parametric studies and incurred the highest computational cost. The Method of Lines (MoL) produced outputs that were consistent with the Backward Euler method in terms of smoothness and stability at a lower computational cost, but occasionally introduced edge artifacts. Thus, choosing the right numerical scheme really comes down to balancing stability, sensitivity to parameters, computational cost, and how well the method preserves visual details during denoising.

7 Conclusion

This study explored the use of the 2D heat equation for image denoising through four numerical schemes to discretize the time evolution: **Forward Euler**, **Backward Euler**, **Crank–Nicolson**, and **Method of Lines (MoL)**. Each method was implemented from scratch, evaluated on synthetic noisy data, and analyzed for its sensitivity to time step, diffusion coefficient, and iteration count. All schemes are theoretically valid and possess known accuracy orders—first-order in time for Forward and Backward Euler, second-order in time for Crank–Nicolson, and fourth-order in time for the RK45-based MoL method. Additionally, centered-difference approximations were used for all spatial derivatives, providing second-order accuracy in space. The study however poses a couple of significant questions which need to be investigated:

1. Why does the Crank–Nicolson method appear to amplify noise rather than suppress it? Could there be some finer stability or consistency condition at play that I might be overlooking?

2. What causes the appearance of dark edge artifacts in certain cases with Backward Euler and the Method of Lines (MoL), while such effects are not observed with Forward Euler?

Furthermore this study focused on qualitative evaluation through visual inspection, incorporating quantitative metrics such as PSNR or SSIM would enable more objective comparisons. Immediate potential extensions also include deriving precise CFL and matrix invertibility conditions, performing detailed accuracy and computational cost analyses, and evaluating performance on real-world image data. While some of these derivations were carried out as part of the rough work, they were omitted from the final report due to space constraints. In general, the project reaffirmed the importance of aligning the numerical schemes with both the mathematical formulation and the goals of the denoising task.