Numerical methods in geophysics

Pr, BECKERS Jean-Marie

E9: Two-dimensional Diffusion

Danish Ali, Owen Dupuy

Table Of Content:

1- Diffusion Modeling in Geophysical Systems:

1.1- Fundamentals of Two-Dimensional Diffusion:

Two-dimensional diffusion is essential for modeling processes like pollutant transport, thermal conduction, and fluid movement in geophysical systems. The diffusion equation, describes how a quantity C (e.g., concentration or temperature) evolves over time, where κ is the diffusion coefficient and S represents sources.

$$\frac{\partial C}{\partial t} = \kappa \left(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2} \right) + S,$$

In pollutant transport, the equation models the spread of contaminants under initial conditions like localized pollution patches and boundary constraints such as no-flux (Neumann) conditions. For thermal diffusion, it predicts heat conduction in rocks, influenced by material properties and external heat sources. Numerical solutions use methods like **explicit** and **implicit finite differences**. The explicit scheme is simpler but requires small time steps for stability, while the implicit scheme allows larger steps but is computationally heavier. Accurate 2D modeling requires balancing grid resolution, stability, and efficiency while considering factors like anisotropic diffusion or heterogeneous media for realistic simulations.

1.2- Applications of Diffusion Processes in Geophysics:

Diffusion processes are fundamental in geophysics for modeling contaminant transport, heat conduction, and seismic wave propagation in Earth's subsurface. In **groundwater flow**, the advection-diffusion equation describes how pollutants spread through aquifers, aiding in predicting contaminant migration and optimizing remediation. For **heat transfer in rocks**, the thermal diffusion equation models heat conduction driven by sources like radioactive decay or magma intrusions, critical for geothermal energy studies and volcanic activity analysis. In **seismic wave propagation**, hyperbolic PDEs simulate wave movement, incorporating diffusion-like attenuation to predict energy loss during earthquakes or subsurface imaging. These applications rely on numerical techniques, such as finite difference methods and implicit schemes, to solve PDEs efficiently while balancing stability, accuracy, and computational cost.

Feature	Groundwater Flow	Heat Transfer	Seismic Propagation
Governing Equation	Advection-Diffusion Equation	Thermal Diffusion Equation	Elastic/Hyperbolic Wave Equation
Physical Process	Solute transport in porous media	Conduction in geological media	Wave propagation and attenuation
Dominant Behavior	Advection + Diffusion	Diffusion	Hyperbolic (wave-like) + damping
Time Scales	Days to years	Hours to millions of years	Seconds to minutes
Numerical Challenges	Stability (CFL), anisotropy	Nonlinear heat sources	High-frequency oscillations, ABC

2- Mathematical Approach:

2.1- Diffusion Equation:

$$\frac{\partial C}{\partial t} = D\left(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}\right) \quad (1)$$

Time derivative, using a forward difference approximation:

$$\frac{\partial C}{\partial t} \approx \frac{C_{i,j}^{n+1} - C_{i,j}^{n}}{\Delta t} \quad (2)$$

2.2- Explicit Scheme:

Spatial Second Derivatives using central difference approximations:

$$\frac{\partial^2 C}{\partial x^2} \approx \frac{C_{i+1,j}^{n} - 2C_{i,j}^{n} + C_{i-1,j}^{n}}{\Delta x^2} \quad (3)$$

Fluxes interactions from left and right at time = n

$$\frac{\partial^2 C}{\partial y^2} \approx \frac{C_{i,j+1}^{n} - 2C_{i,j}^{n} + C_{i,j-1}^{n}}{\Delta y^2} \quad (4)$$

Fluxes interactions from top and bottom at time = n

Implementing in the initial equation:

(1) ⇔ (2) = (3) + (4)

$$\frac{C_{i,j}^{n+1} - C_{i,j}^{n}}{\Delta t} = D\left(\frac{C_{i+1,j}^{n} - 2C_{i,j}^{n} + C_{i-1,j}^{n}}{\Delta x^2} + \frac{C_{i,j+1}^{n} - 2C_{i,j}^{n} + C_{i,j-1}^{n}}{\Delta y^2}\right)$$

Now isolating the term at time = n+1

$$C_{i,j}^{n+1} = C_{i,j}^{n} + \Delta t \cdot D\left(\frac{C_{i+1,j}^{n} - 2C_{i,j}^{n} + C_{i-1,j}^{n}}{\Delta x^2} + \frac{C_{i,j+1}^{n} - 2C_{i,j}^{n} + C_{i,j-1}^{n}}{\Delta y^2}\right)$$

2.3- Implicit Scheme:

Spatial Second Derivatives using central difference approximations:

$$\frac{\partial^2 C}{\partial x^2} \approx \frac{C_{i+1,j}^{n+1} - 2C_{i,j}^{n+1} + C_{i-1,j}^{n+1}}{\Delta x^2} \quad (3)$$

Fluxes interactions from left and right at time = n+1

$$\frac{\partial^2 C}{\partial y^2} \approx \frac{C_{i,j+1}^{n+1} - 2C_{i,j}^{n+1} + C_{i,j-1}^{n+1}}{\Delta y^2} \quad (4)$$

Fluxes interactions from top and bottom at time = n+1

Implementing in the initial equation:

(1) ⇔ (2) = (3) + (4)

$$\frac{C_{i,j}^{n+1} - C_{i,j}^{n}}{\Delta t} = D\left(\frac{C_{i+1,j}^{n+1} - 2C_{i,j}^{n+1} + C_{i-1,j}^{n+1}}{\Delta x^2} + \frac{C_{i,j+1}^{n+1} - 2C_{i,j}^{n+1} + C_{i,j-1}^{n+1}}{\Delta y^2}\right)$$

Now isolating the terms at time = n+1

$$C_{i,j}^{n+1} - \Delta t \cdot D\left(\frac{C_{i+1,j}^{n+1} - 2C_{i,j}^{n+1} + C_{i-1,j}^{n+1}}{\Delta x^2} + \frac{C_{i,j+1}^{n+1} - 2C_{i,j}^{n+1} + C_{i,j-1}^{n+1}}{\Delta y^2}\right) = C_{i,j}^{n}$$

We can write this as a linear system using matrixes: $A \cdot C^{n+1} = C^n$

- A: Coefficient matrix describing the diffusion operator.
- C^{n+1}: Unknown vector containing $C_{i,j}^{n+1}$ at all grid points.
- C^n: Known vector containing $C_{i,j}^{n}$ at all grid points.

Let’s consider a 2D grid of Nx×Ny. The total number of grid points is N=Nx×Ny. We'll map the 2D indices (i,j) into a **1D index**: k=(i−1)·Ny+j

$$A = \begin{bmatrix} \alpha & \beta & 0 & \cdots & \gamma & 0 & \cdots & 0 \\ \beta & \alpha & \beta & 0 & \cdots & \gamma & \cdots & 0 \\ 0 & \beta & \alpha & \beta & 0 & \cdots & \gamma & 0 \\ \vdots & \vdots & \beta & \alpha & \beta & 0 & \cdots & \gamma \\ \gamma & \cdots & 0 & \beta & \alpha & \beta & 0 & \cdots \\ 0 & \gamma & \cdots & 0 & \beta & \alpha & \beta & \cdots \\ \vdots & \cdots & \gamma & 0 & \cdots & \beta & \alpha & \beta \\ 0 & \cdots & \gamma & 0 & \cdots & 0 & \beta & \alpha \end{bmatrix}$$

- $\alpha = 1 + 2D\Delta t \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right)$ (central coefficient)
- $\beta = -D\Delta t / \Delta y^2$ (vertical neighbors)
- $\gamma = -D\Delta t / \Delta x^2$ (horizontal neighbors)

2.4- Time discretization:

Aspect	Explicit Scheme	Implicit Scheme
Time Derivative	Forward difference ($n \to n+1$)	Backward difference ($n+1$)
Stability	Conditionally stable (CFL condition)	Unconditionally stable
Time Step (Δt)	Small Δt required for stability	Large Δt can be used
Complexity	Simple, no matrix inversion	Complex, requires solving a linear system
Accuracy	Good for small Δt	Less accurate for large Δt
Computational Cost	Low per step, but many steps needed	High per step, but fewer steps needed

Courant–Friedrichs–Lewy (CFL) condition: $$\Delta t \leq \frac{\min(\Delta x^2, \Delta y^2)}{4D}$$

3- Problem Statement: 2D diffusion of a patch of pollution

3.1- Case Study:

Two-dimensional diffusion: $$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(\kappa \frac{\partial C}{\partial x}\right) + \frac{\partial}{\partial y}\left(\kappa \frac{\partial C}{\partial y}\right)$$

In a domain $x \in [0,L]$, $y \in [0,L]$. L=1000 m, κ =1 $m2/s$

Initial condition: patch of pollution
$C(x, y, t = 0) = 10$ if $(x-L/4)^2+(y-L/4)^2 < (L/4)^2$
$C(x, y, t = 0) = 0$ elsewhere

Boundary condition in y: $\partial C/\partial y = 0$ in $y = 0,L$
Boundary condition in x: $\partial C/\partial x = 0$ in $x = 0,L$

3.2- Initial conditions:

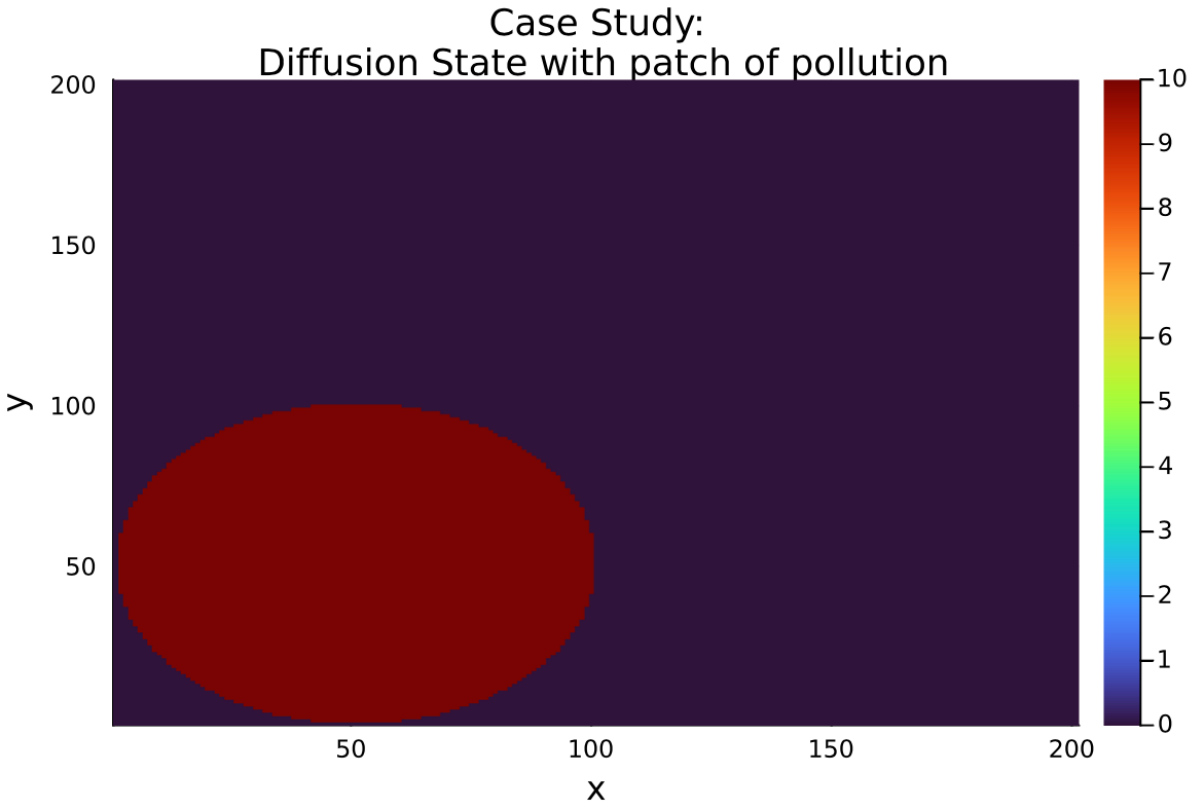


Figure 1: Initial Concentration Field for 2D Diffusion of a Patch of Pollution (Number of grid points in x and y directions: Nx, Ny = 201, 201)

4- Boundary Condition Effects in the Current Study

4.1- Neumann boundary conditions :

The current study on two-dimensional diffusion of a patch of pollution uses **Neumann boundary conditions** ($\partial C/\partial x=0$and $\partial C/\partial y=0$) at all boundaries. These conditions imply that no flux of pollutant crosses the domain boundaries, representing a **closed system** where the total concentration remains conserved. Physically, this corresponds to a scenario where the pollution is trapped, and no material escapes or enters the system. This assumption aligns with the problem setup and is critical for maintaining mass conservation over time.

The Neumann conditions are well-suited for this type of problem as they ensure that the solution evolves realistically within the domain without introducing artificial losses or gains at the boundaries. From a numerical perspective, they are implemented by approximating the derivatives at the boundaries using finite differences. Proper handling of these conditions is crucial because errors in discretizing the boundary gradients could propagate into the interior, affecting the overall accuracy of the solution.

4.2- Impact on explicit and implicit scheme :

The imposed **Neumann boundary conditions** ($\partial C/\partial x=0$ and $\partial C/\partial y=0$) are expected to impact the behavior of both the **explicit** and **implicit schemes** in a similar way, particularly near the domain boundaries. Since these conditions enforce zero flux, they ensure that no concentration is lost or gained at the edges, preserving the total mass within the system.

For the **explicit scheme**, the Neumann conditions are sensitive to stability constraints. If the CFL condition is not met, errors near the boundaries can amplify and propagate inward. When the time step is appropriately chosen, the scheme handles these conditions accurately, preserving mass conservation. In the **implicit scheme**, the unconditional stability ensures robust enforcement of the Neumann boundaries even with larger time steps. However, the smoothing effect of the implicit scheme may slightly reduce accuracy near the boundaries by dampening sharp gradients.

4.3- Role of Neumann boundary conditions :

While other boundary conditions, such as **Dirichlet** or **Robin**, could alter the behavior by imposing fixed values or flux exchanges, they are not relevant here. The imposed Neumann conditions appropriately model the no-flux scenario, ensuring total pollutant conservation and providing a solid foundation for analyzing the numerical performance of both schemes.

In summary, the imposed Neumann boundary conditions ensure that the total pollutant concentration is conserved within the domain, which is critical for accurate modeling of closed systems. The results obtained with these conditions reflect the expected behavior of the diffusion process and provide a reliable foundation for analyzing the numerical performance of the explicit and implicit schemes.

5- Time Discretisation

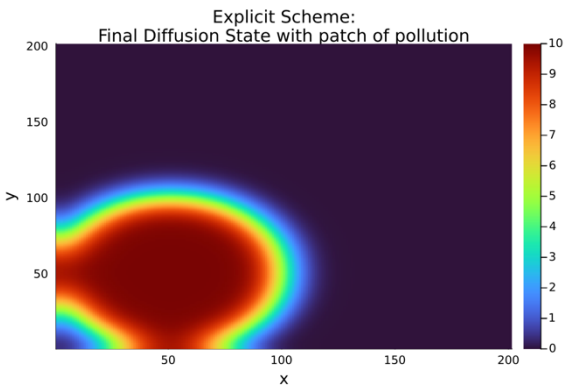
5.1-Explicit Scheme :

Parameters:

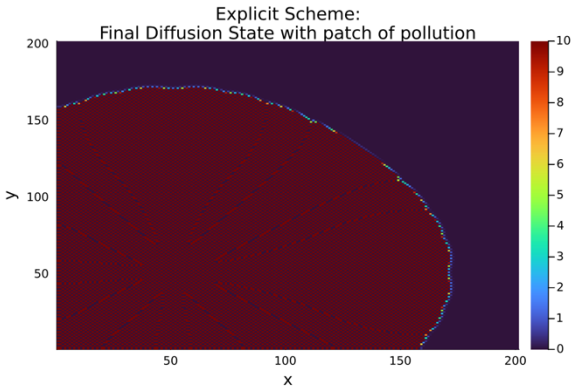
Domain size	L = 1000.0
Number of grid points in x and y directions	Nx, Ny = 201, 201
Diffusion coefficient	kappa = 1.0
Total simulation time	T = 1000.0

CFL condition : dt must be < 6.25

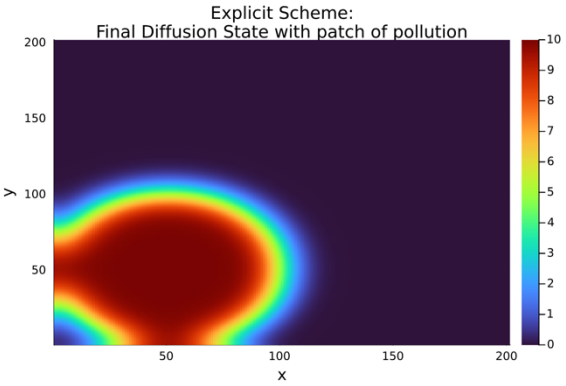
Time step: dt = 10
Compute time: 5.22 s
Initial total concentration: 78250.0
Final total concentration: 1.29e30
Compute error: 1.29e30
% error: 1.65e27 %



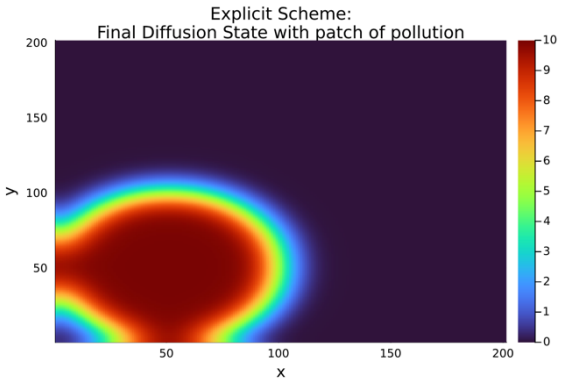
Time step: dt = 5
Compute time: 9.99 s
Initial total concentration: 78250.0
Final total concentration: 79146.5
Compute error: 896.5
% error: 1.15 %



Time step: dt = 1
Compute time: 47.72 s
Initial total concentration: 78250.0
Final total concentration: 79212.4
Compute error: 962.4
% error: 1.23 %



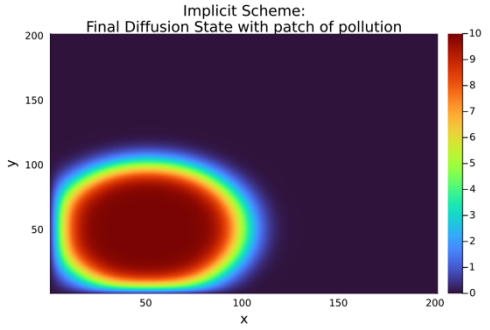
Time step: dt = 0,1
Compute time: 7min 21.80 s
Initial total concentration: 78250.0
Final total concentration: 79227.2
Compute error: 977.2
% error: 1.25 %



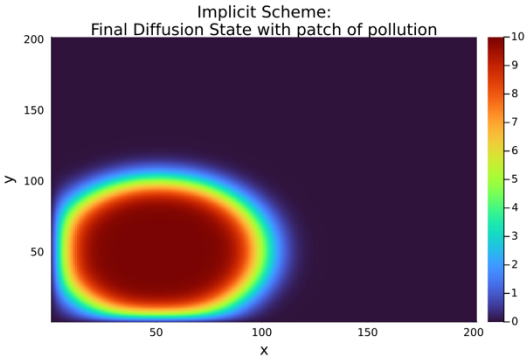
5.2-Implicit Scheme :

Parameters:

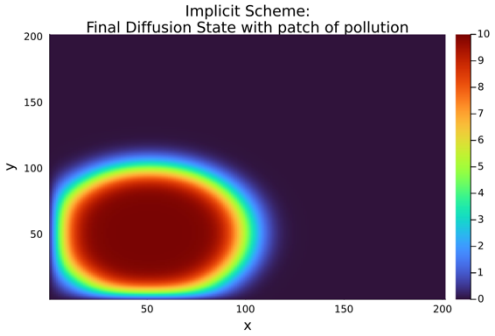
Domain size	L = 1000.0
Number of grid points in x and y directions	Nx, Ny = 201, 201
Diffusion coefficient	kappa = 1.0
Total simulation time	T = 1000.0



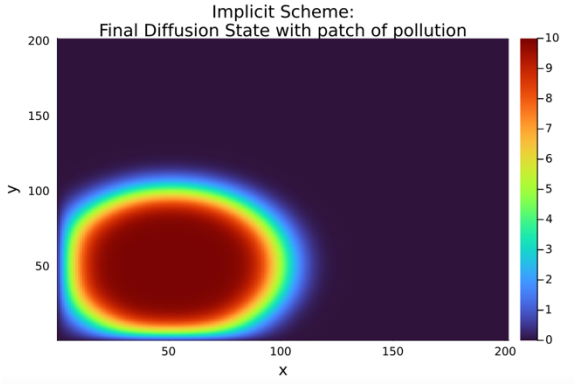
Time step: dt = 10
Compute time: 3.17 s
Initial total concentration: 78250.0
Final total concentration: 73351.1
Compute error: 4898.9
% error: 6.26 %



Time step: dt = 5
Compute time: 5.44 s
Initial total concentration: 78250.0
Final total concentration: 73348.4
Compute error: 4901.6
% error: 6.26 %



Time step: dt = 1
Compute time: 22.32 s
Initial total concentration: 78250.0
Final total concentration: 73346.2
Compute error: 4903.8
% error: 6.27 %



Time step: dt = 0,1
Compute time: 3 min 28.11 s
Initial total concentration: 78250.0
Final total concentration: 73345.4
Compute error: 4904.6
% error: 6.27 %

5.3-Interpretations for time discretisation :

The **explicit scheme** is conditionally stable and requires the time step dt to satisfy the CFL condition dt<6.25. Larger time steps (e.g., dt=10) lead to instability and unrealistic results. In contrast, the **implicit scheme** is unconditionally stable, allowing larger time steps without instability. However, the implicit scheme shows consistent errors (~6.26%) regardless of the time step, whereas the explicit scheme achieves higher accuracy with smaller dt(errors ~1.23–1.25%).

About the computation cost, in the **explicit scheme**, compute time increases significantly as dtdt decreases: from 5.22 seconds for dt=10 to over 7 minutes for dt=0.1. The **implicit scheme** is computationally more efficient for larger time steps, completing in 3.17 seconds for dt=10 and remaining manageable even at dt=0.1 (3 min 28.11 s).

The **explicit scheme** offers high accuracy for small dtdt but becomes unstable for large time steps and computationally expensive for small ones. The **implicit scheme** ensures stability for any dtdt and is computationally efficient for larger time steps but lacks accuracy improvement with time step refinement. The choice depends on balancing stability, accuracy, and computational cost.

The time step dt=5 offers an ideal balance of **stability**, **accuracy**, and **efficiency** for both schemes. For the **explicit scheme**, it satisfies the CFL condition, ensures stability, and achieves a low error of **1.15%** with a reasonable compute time of **9.99 seconds**. Smaller time steps improve accuracy only slightly but significantly increase computational cost. For the **implicit scheme**, dt=5 minimizes compute time (**5.44 seconds**) while maintaining consistent error (**6.26%**), as reducing dt further does not improve accuracy.

We will proceed with dt=5 for both schemes in the next phase on **space discretization** to ensure reliable results with efficient computation.

Time Step (dt)	Explicit Scheme - Error (%)	Implicit Scheme - Error (%)	Explicit Scheme - Compute Time (s)	Implicit Scheme - Compute Time (s)
10	1,65E+27	6,26	5,22	3,17
5	1,15	6,26	9,99	5,44
1	1,23	6,27	47,72	22,32
0,1	1,25	6,27	441,8	208,11

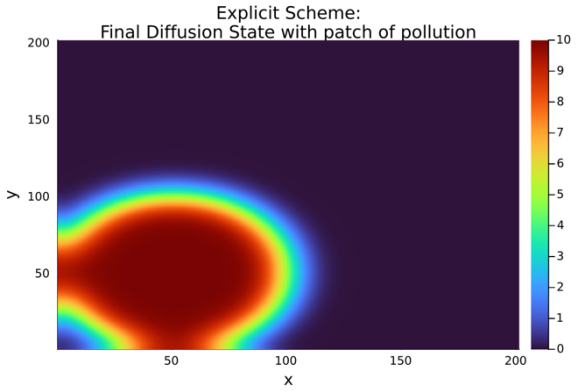
6- Space Discretisation

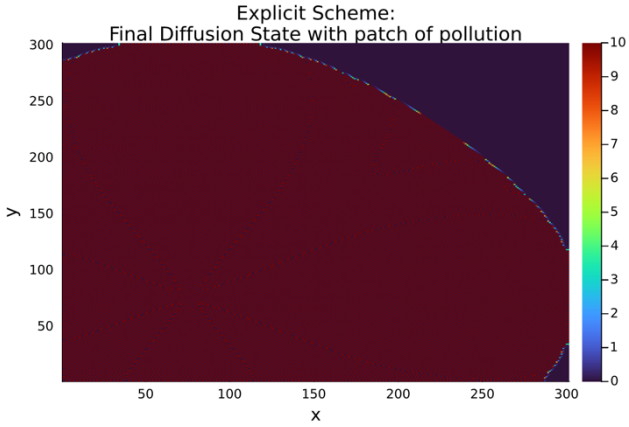
6.1-Explicit Scheme :

Parameters:

Domain size	L = 1000.0
Time step	dt = 5
Diffusion coefficient	kappa = 1.0
Total simulation time	T = 1000.0

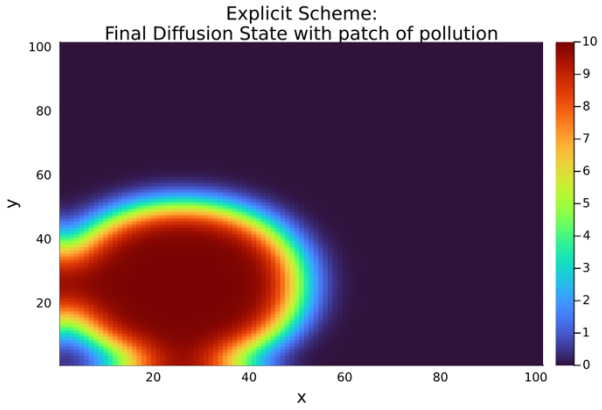
Number of grid points in x and y directions :
Nx, Ny = 201, 201
Compute time: 9.80 s
Initial total concentration: 78250.0
Final total concentration: 79146.5
Compute error: 896.5
% error: 1.15 %
CFL condition : dt must be < 6.25

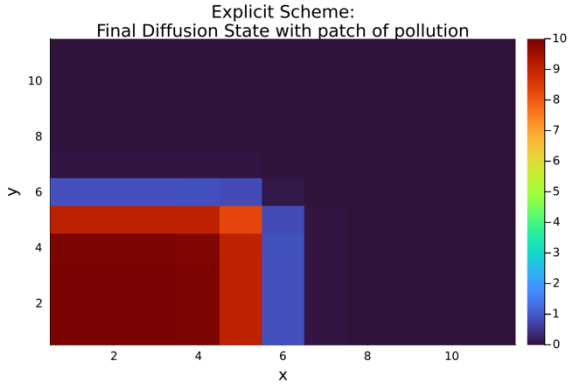




Number of grid points in x and y directions :
Nx, Ny = 301, 301
Compute time: 21.61 s
Initial total concentration: 176450.0
Final total concentration: -3.41e77
Compute error: 3.41e77
% error: 1.93e74 %
CFL condition : dt must be < 2.78

Number of grid points in x and y directions :
Nx, Ny = 101, 101
Compute time: 2.68 s
Initial total concentration: 19410.0
Final total concentration: 19899.4
Compute error: 489.4
% error: 2.52 %
CFL condition : dt must be < 25.0





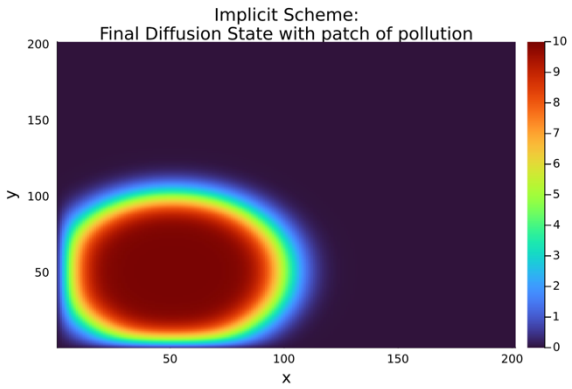
Number of grid points in x and y directions :
Nx, Ny = 11, 11
Compute time: 0.02 s
Initial total concentration: 160.0
Final total concentration: 249.9
Compute error: 89.9
% error: 56.19 %
CFL condition : dt must be < 2500.0

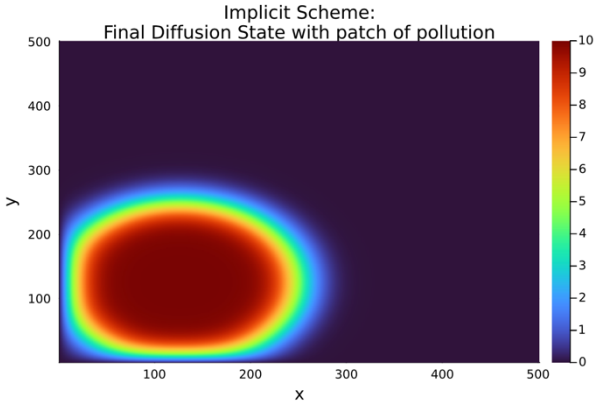
6.2-Implicit Scheme :

Parameters:

Domain size	L = 1000.0
Time step	dt = 5
Diffusion coefficient	kappa = 1.0
Total simulation time	T = 1000.0

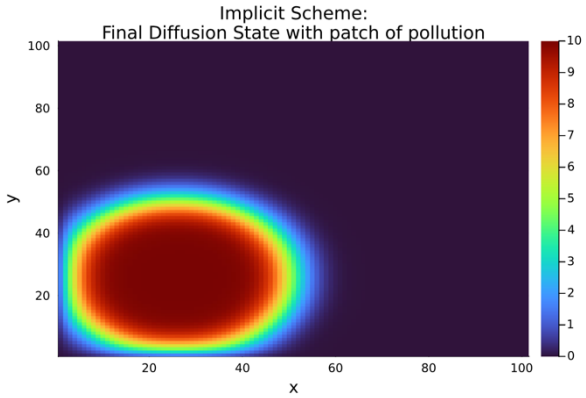
Number of grid points in x and y directions :
Nx, Ny = 201, 201
Compute time: 5.67 s
Initial total concentration: 78250.0
Final total concentration: 73348.4
Compute error: 4901.6
% error: 6.26 %

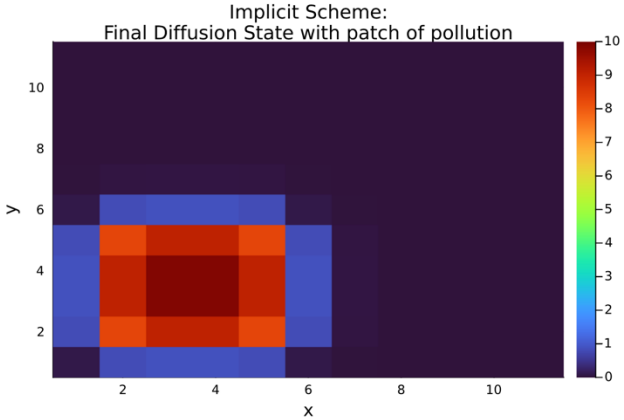




Number of grid points in x and y directions :
Nx, Ny = 501, 501
Compute time: 1 min 14.69 s
Initial total concentration: 490490.0
Final total concentration: 456140.2
Compute error: 34349.8
% error: 7.0 %

Number of grid points in x and y directions :
Nx, Ny = 101, 101
Compute time: 1.38 s
Initial total concentration: 19410.0
Final total concentration: 18422.0
Compute error: 988.0
% error: 5.09 %





Number of grid points in x and y directions :
Nx, Ny = 11, 11
Compute time: 0.02 s
Initial total concentration: 160.0
Final total concentration: 159.6
Compute error: 0.4
% error: 0.23 %

6.3-Interpretations for space discretisation :

The **explicit scheme** is more sensitive to the grid size because of its reliance on the CFL condition for stability. As the grid resolution increases, the time step must be reduced to maintain stability, which increases computational cost. For example, at **Nx, Ny = 301, 301**, the explicit scheme becomes unstable because dt=5dt=5 is too large for the refined grid.

The **implicit scheme**, on the other hand, is unconditionally stable, allowing larger grid resolutions without requiring time step adjustments. This makes the implicit scheme more robust for higher grid sizes. However, the implicit scheme does not show significant accuracy improvements with finer grids, and the computational cost increases substantially at very high resolutions, such as **Nx, Ny = 501, 501**.

At lower grid resolutions (**Nx, Ny = 11, 11**), the implicit scheme outperforms the explicit scheme with much lower error (**0.23% vs. 56.19%**). At medium grid resolutions (**Nx, Ny = 201, 201**), the explicit scheme achieves higher accuracy (**1.15%**) compared to the implicit scheme (**6.26%**) but at a higher computational cost.

For both schemes, **Nx, Ny = 201, 201** provides the best balance between accuracy, stability, and computational cost. The **explicit scheme** achieves slightly better accuracy but requires careful adherence to the CFL condition. The **implicit scheme** ensures stability without additional time step restrictions but has a higher error that does not improve significantly with grid refinement.

We will proceed with **Nx, Ny = 201, 201** for both the explicit and implicit schemes in further studies. This resolution ensures reliable results while maintaining a manageable computational cost.

Grid Size (Nx, Ny)	Explicit Scheme - Error (%)	Implicit Scheme - Error (%)	Explicit Scheme - Compute Time (s)	Implicit Scheme - Compute Time (s)
301, 301	1,93E+74	7	21,61	74,69
201, 201	1,15	6,26	9,8	5,67
101, 101	2,52	5,09	2,68	1,38
11, 11	56,19	0,23	0,02	0,02

7- Total simulation time

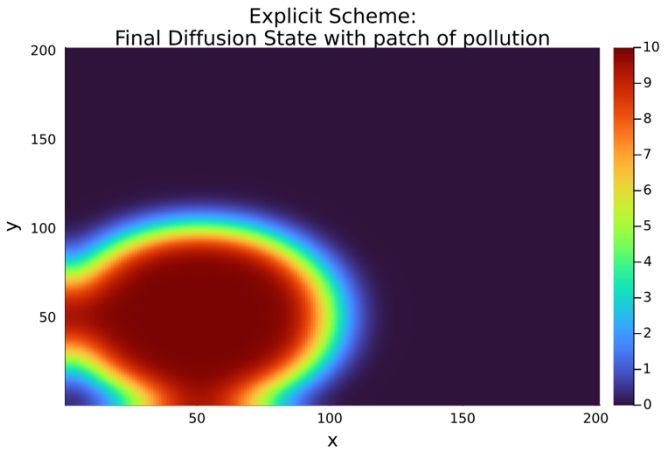
7.1-Explicit Scheme :

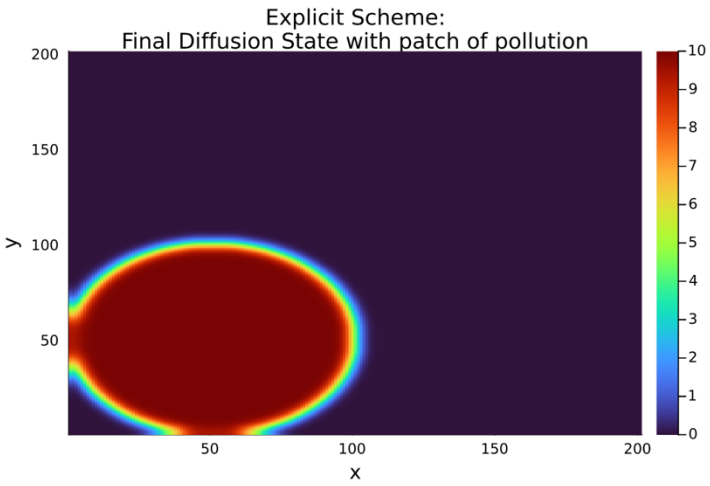
Parameters:

Domain size	L = 1000.0
Time step	dt = 5
Number of grid points in x and y directions :	Nx, Ny = 201, 201
Diffusion coefficient	kappa = 1.0

CFL condition : dt must be < 6.25

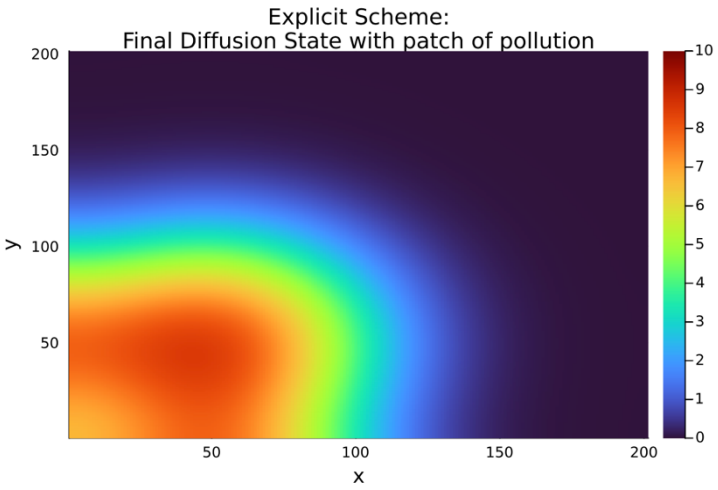
Total simulation time: T = 1 000 s
Compute time: 9.80 s
Initial total concentration: 78250.0
Final total concentration: 79146.5
Compute error: 896.5
% error: 1.15 %





Total simulation time: T = 100 s
Compute time: 1.11 s
Initial total concentration: 78250.0
Final total concentration: 78774.0
Compute error: 524.0
% error: 0.67 %

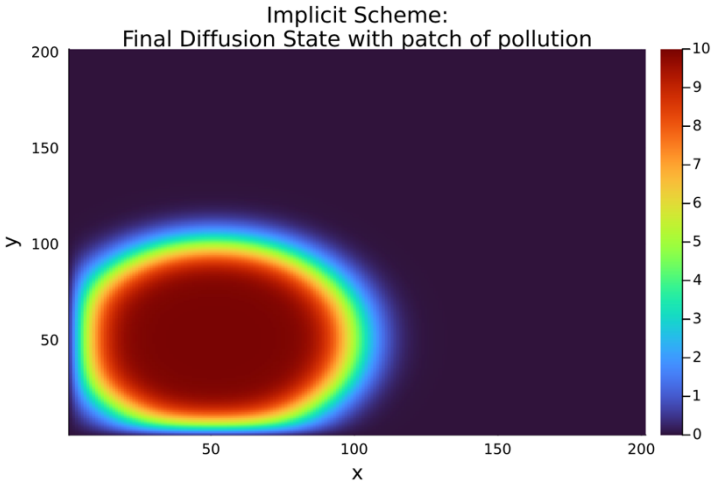
Total simulation time: T = 10 000 s
Compute time: 85.17 s
Initial total concentration: 78250.0
Final total concentration: 79649.6
Compute error: 1399.6
% error: 1.79 %



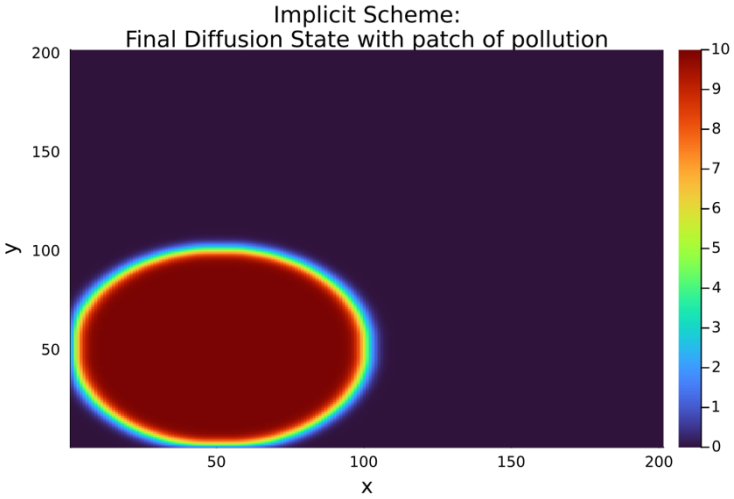
7.2-Implicit Scheme :

Parameters:

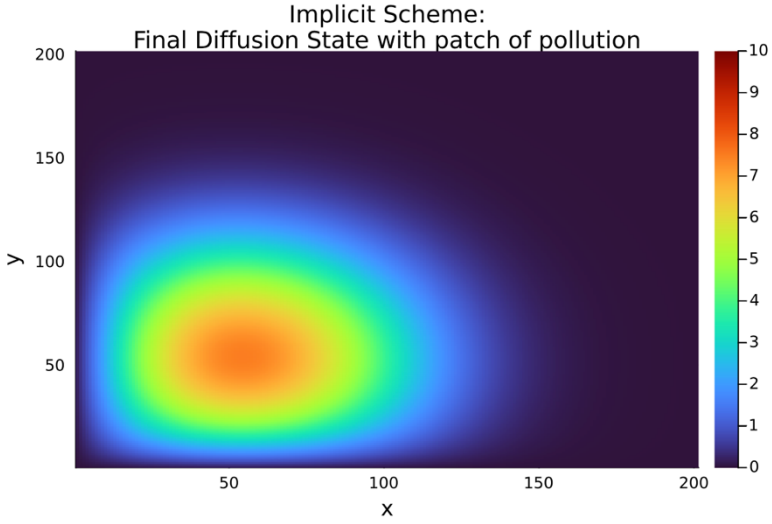
Domain size	L = 1000.0
Time step	dt = 5
Number of grid points in x and y directions :	Nx, Ny = 201, 201
Diffusion coefficient	kappa = 1.0



Total simulation time: T = 1 000 s
Compute time: 6.41 s
Initial total concentration: 78250.0
Final total concentration: 73348.4
Compute error: 4901.6
% error: 6.26 %



Total simulation time: T = 100 s
Compute time: 1.78 s
Initial total concentration: 78250.0
Final total concentration: 77682.3
Compute error: 567.7
% error: 0.73 %



Total simulation time: T = 10 000 s
Compute time: 43.48 s
Initial total concentration: 78250.0
Final total concentration: 52288.8
Compute error: 25961.2
% error: 33.18 %

7.3-Scheme stability over time :

The impact of total simulation time shows clear differences between the explicit and implicit schemes in terms of accuracy and computational cost.

In the **explicit scheme**, shorter simulation times yield better accuracy and lower compute times. At **T = 100 s**, the error is very low (**0.67%**) with a short compute time. For **T = 1,000 s**, the error rises slightly to **1.15%**, while compute time increases to **9.80 seconds**, maintaining a good balance. At **T = 10,000 s**, accuracy decreases and compute time grows significantly. This indicates that while the explicit scheme maintains good accuracy over short simulation times, it becomes more computationally expensive and slightly less accurate for extended total times.

In the **implicit scheme**, compute times are consistently lower, but accuracy declines over longer simulations. At **T = 100 s**, the error is **0.73%**, with low compute time. At **T = 1,000 s**, the error increases to **6.26%**, while compute time remains efficient at **6.41 seconds**. However, for **T = 10,000 s**, accuracy deteriorates significantly (**33.18% error**) with larger compute time. The compute time remains shorter than the explicit scheme at **43.48 seconds**, but the loss in accuracy at longer total times highlights a drawback of the implicit scheme.

The best total simulation time is **T = 1,000 s**, providing a good balance between accuracy and computational cost for both schemes. The explicit scheme achieves lower errors, while the implicit scheme remains computationally efficient, making this time ideal for observing meaningful diffusion behavior without excessive computation.

Total Simulation Time (s)	Explicit Scheme - Error (%)	Implicit Scheme - Error (%)	Explicit Scheme - Compute Time (s)	Implicit Scheme - Compute Time (s)
100	0,67	0,73	1,11	1,78
1000	1,15	6,26	9,8	6,41
10000	1,79	33,18	85,17	43,48

8- Disscusion

8.1- Comparative Analysis of Explicit and Implicit Schemes:

Both the explicit and implicit schemes demonstrate distinct advantages and limitations, which must be considered based on the specific application and computational resources available. The **explicit scheme**, while straightforward to implement, is highly sensitive to the Courant-Friedrichs-Lewy (CFL) condition, which imposes restrictions on the time step size for stability. This makes it computationally expensive for fine grid resolutions or long simulations, but it offers higher accuracy for small time steps. In contrast, the **implicit scheme** ensures unconditional stability, allowing for larger time steps and efficient simulations, but it introduces a smoothing effect that reduces accuracy, particularly for sharp gradients or short time-scale phenomena.

8.2- Accuracy, Stability, and Computational Efficiency:

The implementation of **Neumann boundary conditions** in this study highlights their role in maintaining mass conservation in closed systems, ensuring that no artificial flux is introduced at the domain boundaries. However, inaccuracies in handling boundary derivatives can propagate into the interior solution, particularly for the explicit scheme, where errors can amplify under unstable conditions.

The **space and time discretization** results reveal that grid resolution and time step size must be carefully balanced. While finer grids improve spatial accuracy, they also increase computational costs and stability constraints for the explicit scheme. The implicit scheme, though more robust to grid refinement, does not significantly improve in accuracy at higher resolutions, indicating diminishing returns for additional computational effort.

Furthermore, the comparison of total **simulation times** demonstrates that both schemes perform well for shorter simulations, but accuracy and efficiency decline over extended time scales. For practical applications, such as pollutant transport or heat diffusion, achieving an optimal balance between accuracy, stability, and computational cost is critical.

9- Conclusion

This study highlights the significance of numerical methods in modeling diffusion processes, particularly in two-dimensional geophysical systems. By solving the diffusion equation for applications like pollutant transport we gain insights into how physical properties evolve over time and space.

The comparison between **explicit** and **implicit schemes** illustrates key trade-offs: while the explicit method offers higher accuracy for smaller time steps, it is conditionally stable and computationally demanding as grid resolution increases. In contrast, the implicit method ensures stability for larger time steps but sacrifices some accuracy. Through careful spatial and temporal discretization, a balance can be achieved between computational efficiency, numerical stability, and solution accuracy.

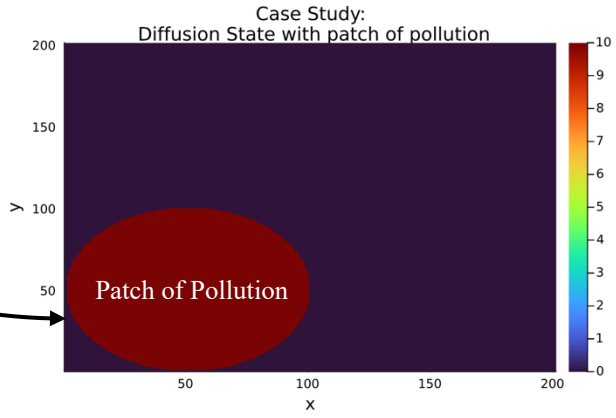
The results underscore the importance of proper boundary conditions, such as the Neumann no-flux condition, which ensures realistic modeling of closed systems. Additionally, optimal grid resolution and time step selection are critical to achieving reliable simulations without excessive computational costs.

In conclusion, numerical modeling of 2D diffusion provides a robust framework for solving complex geophysical problems. By selecting appropriate schemes and parameters, these models serve as powerful tools for predicting and understanding key processes in groundwater flow, heat transfer, and seismic wave propagation. Future work can incorporate anisotropic diffusion, heterogeneous media, and advanced solvers to further enhance model accuracy and applicability.

APPENDIX - Julia Code

APPENDIX.1- Explicit Scheme's Code:

```julia
@time begin
using Pkg
using Plots

# Parameters
L = 1000.0          # Domain size
Nx, Ny = 201, 201   # Number of grid points in x and y directions
kappa = 1.0         # Diffusion coefficient
dx = L / (Nx - 1)   # Grid spacing in x
dy = L / (Ny - 1)   # Grid spacing in y
dt = 5              # Time step
T = 1000.0          # Total simulation time
nt = Int(T / dt)    # Number of time steps

# Initialize the concentration field
Cn = zeros(Nx, Ny)

# Initial condition: patch of pollution
for i in 1:Nx
    for j in 1:Ny
        x = (i - 1) * dx
        y = (j - 1) * dy
        if (x - L/4)^2 + (y - L/4)^2 < (L/4)^2
            Cn[i, j] = 10.0
        end
    end
end

# Compute the sum of the initial concentration field
initial_sum = sum(Cn)

# Time-stepping loop
for t in 1:nt
    # Compute Central Difference Approximation
    d2Cdx2 = zeros(Nx, Ny)
    d2Cdy2 = zeros(Nx, Ny)

    for i in 2:Nx-1
        for j in 2:Ny-1
            d2Cdx2[i, j] = (Cn[i+1, j] - 2*Cn[i, j] + Cn[i-1, j]) / dx^2
            d2Cdy2[i, j] = (Cn[i, j+1] - 2*Cn[i, j] + Cn[i, j-1]) / dy^2
        end
    end

    # Implementing in the 2D Diffusion equation
    Cnp = Cn .+ dt * kappa .* (d2Cdx2 + d2Cdy2)

    # Apply boundary conditions, all equals to 0
    Cnp[1, :] .= Cnp[2, :]
    Cnp[end, :] .= Cnp[end-1, :]
    Cnp[:, 1] .= Cnp[:, 2]
    Cnp[:, end] .= Cnp[:, end-1]

    # Update the field
    Cn = Cnp

end
end
# Compute the sum of the final concentration field
final_sum = sum(Cn)
println("Initial total concentration: ",round(initial_sum; digits=1))
println("Final total concentration: ", round(final_sum; digits=1))
compute_error = final_sum - initial_sum
println("Compute error: ", round(abs(compute_error); digits=1))
println("% error: ", round(abs(compute_error) * 100 / initial_sum; digits=2), " %")
println("CFL condition : dt must be < ", round(min(dx^2, dy^2) / (4 * kappa); digits=10))

# Final plot

heatmap(Cn, xlabel="x", ylabel="y",
    title="Explicit Scheme:\nFinal Diffusion State with patch of pollution",
    color=:turbo, clim=(0, 10),titlefontsize=12)
```



$$C_{i,j}^{n+1} = C_{i,j}^{n} + \Delta t \cdot D \left(\frac{C_{i+1,j}^{n} - 2C_{i,j}^{n} + C_{i-1,j}^{n}}{\Delta x^2} + \frac{C_{i,j+1}^{n} - 2C_{i,j}^{n} + C_{i,j-1}^{n}}{\Delta y^2} \right)$$

APPENDIX.2- Implicit Scheme's Code:

```
@time begin
using Pkg
using SparseArrays
using Plots
using IterativeSolvers

# Parameters
L = 1000.0           # Domain size
Nx, Ny = 201, 201    # Number of grid points in x and y directions
kappa = 1.0          # Diffusion coefficient
dx = L / (Nx - 1)    # Grid spacing in x
dy = L / (Ny - 1)    # Grid spacing in y
dt = 5               # Time step
T = 1000.0           # Total simulation time
nt = Int(T / dt)     # Number of time steps

# Initialize the concentration field
Cn = zeros(Nx, Ny)

# Initial condition: patch of pollution
for i in 1:Nx
    for j in 1:Ny
        x = (i - 1) * dx
        y = (j - 1) * dy
        if (x - L/4)^2 + (y - L/4)^2 < (L/4)^2
            Cn[i, j] = 10.0
        end
    end
end

# Assemble the implicit scheme's matrix (A) related to Ci,j ; Ci+1,j ; Ci-1,j ; Ci,j+1 ; Ci,j-1
N = Nx * Ny                  # Total number of grid points
#A = zeros(N, N)              # Full coefficient matrix (non-sparse)

################################################ Alternative to go faster
# Replace the dense matrix with a sparse matrix
A = spzeros(N, N)
################################################

for i in 1:Nx
    for j in 1:Ny
        idx = (i - 1) * Ny + j  # Map 2D index (i, j) to 1D index

        # Diagonal element Ci,j
        A[idx, idx] = 1 + 2 * kappa * dt * (1 / dx^2 + 1 / dy^2)

        # Neighbors in x-direction Ci+1,j ; Ci-1,j
        if i > 1
            A[idx, idx - Ny] = -kappa * dt / dx^2
        end
        if i < Nx
            A[idx, idx + Ny] = -kappa * dt / dx^2
        end

        # Neighbors in y-direction Ci,j+1 ; Ci,j-1
        if j > 1
            A[idx, idx - 1] = -kappa * dt / dy^2
        end
        if j < Ny
            A[idx, idx + 1] = -kappa * dt / dy^2
        end
    end
end
```

A dense matrix (zeros(N, N)) allocates memory for all N^2 elements, even though most are zeros. For large grids (e.g., Nx,Ny=201, 201), this leads to excessive memory usage. A sparse matrix (spzeros(N, N)) only stores the non-zero elements, significantly reducing memory consumption and then reduce computation time for implicit schemes, which require solving linear systems at each time step.

$$A = \begin{bmatrix} \alpha & \beta & 0 & \cdots & \gamma & 0 & \cdots & 0 \\ \beta & \alpha & \beta & 0 & \cdots & \gamma & \cdots & 0 \\ 0 & \beta & \alpha & \beta & 0 & \cdots & \gamma & 0 \\ \vdots & \vdots & \beta & \alpha & \beta & 0 & \cdots & \gamma \\ \gamma & \cdots & 0 & \beta & \alpha & \beta & 0 & \cdots \\ 0 & \gamma & \cdots & 0 & \beta & \alpha & \beta & \cdots \\ \vdots & \cdots & \gamma & 0 & \cdots & \beta & \alpha & \beta \\ 0 & \cdots & \gamma & 0 & \cdots & 0 & \beta & \alpha \end{bmatrix}$$

- $\alpha = 1 + 2D\Delta t \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)$ (central coefficient)
- $\beta = -D\Delta t/\Delta y^2$ (vertical neighbors)
- $\gamma = -D\Delta t/\Delta x^2$ (horizontal neighbors)

```julia
# Compute the sum of the initial concentration field
initial_sum = sum(Cn)

# Time-stepping loop
for t in 1:nt
    # Moving from 2D to 1D
    Cnn = zeros(N)    #Cnn is the C at time step n but in the calculation we can't also name it Cn
    for i in 1:Nx
        for j in 1:Ny
            idx = (i - 1) * Ny + j
            Cnn[idx] = Cn[i, j]
        end
    end

    # Solve the linear system A * Cnp = Cnn
    #Cnp = A \ Cnn

################################################ Alternative to achieve perfection
    using IterativeSolvers

    # Solve the linear system using CG solver
    Cnp = cg(A, Cnn; abstol=1e-8, maxiter=500)
################################################

    # Comeback to 2D
    for i in 1:Nx
        for j in 1:Ny
            idx = (i - 1) * Ny + j
            Cn[i, j] = Cnp[idx]
        end
    end
end
end
# Compute the sum of the final concentration field
final_sum = sum(Cn)
println("Initial total concentration: ",round(initial_sum; digits=1))
println("Final total concentration: ", round(final_sum; digits=1))
compute_error = final_sum - initial_sum
println("Compute error: ", round(abs(compute_error); digits=1))
println("% error: ", round(abs(compute_error) * 100 / initial_sum; digits=2), " %")

# Final plot
heatmap(Cn, xlabel="x", ylabel="y",
    title="Implicit Scheme:\nFinal Diffusion State with patch of pollution",
    color=:turbo, clim=(0, 10),titlefontsize=12)
```

Iterative solvers like the Conjugate Gradient (CG) method are more efficient for large, sparse systems, as they avoid costly matrix factorization and reduce memory usage. They scale better than direct solvers and allow adjustable precision, making them ideal for large-scale 2D diffusion problems.

```
  5.324104 seconds (66.28 M allocations: 1.558 GiB, 3.78% gc time)
Initial total concentration: 78250.0
Final total concentration: 73348.4
Compute error: 4901.6
% error: 6.26 %
```