1. 方程描述

反应扩散方程为:

$$
\begin{aligned}
&\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{1}{\epsilon} u(1-u^2), \quad (x,y) \in \Omega, t > 0 \\
&u|_{\partial\Omega} = -1 \\
&u|_{t=0} = \begin{cases} 1 & \text{if } (x,y) \in \widetilde{\Omega} \\ -1 & \text{if } (x,y) \notin \widetilde{\Omega} \end{cases}
\end{aligned}
\tag{1}
$$

其中 $\Omega = [-1,1] \times [-1,1]$，且 $\widetilde{\Omega}$ 是一个椭圆区域。

2. 差分格式

2.1 时间离散化

采用隐式的时间离散化方法，例如向后欧拉法，这样可以保证无条件稳定性。时间离散化后，方程可以写为:

$$
\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \frac{\partial^2 u_{i,j}^{n+1}}{\partial x^2} + \frac{\partial^2 u_{i,j}^{n+1}}{\partial y^2} + \frac{1}{\epsilon} u_{i,j}^{n+1}(1 - (u_{i,j}^{n+1})^2) \tag{2}
$$

即

$$
u_{i,j}^{n+1} - \Delta t \left(\frac{\partial^2 u_{i,j}^{n+1}}{\partial x^2} + \frac{\partial^2 u_{i,j}^{n+1}}{\partial y^2} + \frac{1}{\epsilon} u_{i,j}^{n+1}(1 - (u_{i,j}^{n+1})^2) \right) = u_{i,j}^{n} \tag{3}
$$

2.2 空间离散化

采用二阶中心差分法对空间导数进行离散化，得到:

$$
\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} \tag{4}
$$

$$
\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} \tag{5}
$$

将其代入后得到:

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

2.3 离散化方程组

整理后可以得到一个稀疏线性方程组：

$$\begin{aligned}&\left(1+2\Delta t\left(\frac{1}{(\Delta x)^2}+\frac{1}{(\Delta y)^2}\right)+\Delta t\frac{1}{\epsilon}(1-(u_{i,j}^{n+1})^2)\right)u_{i,j}^{n+1}\\&-\Delta t\left(\frac{u_{i+1,j}^{n+1}+u_{i-1,j}^{n+1}}{(\Delta x)^2}+\frac{u_{i,j+1}^{n+1}+u_{i,j-1}^{n+1}}{(\Delta y)^2}\right)=u_{i,j}^{n}\end{aligned}\tag{7}$$

3. 数值求解

为了求解上述线性方程组，可以使用迭代方法，例如 Gauss-Seidel 方法或共轭梯度法。边界条件和初值条件可以根据题目中的要求进行设置。

4. 程序实现

```python
import numpy as np
import matplotlib.pyplot as plt

# 参数设置
epsilon = 0.01
a = 0.5
b = 0.5
L = 2
T = 0.1
Nx = 50
Ny = 50
dt = 0.0001
dx = L / (Nx - 1)
dy = L / (Ny - 1)
Nt = int(T / dt)

# 初始化网格
x = np.linspace(-1, 1, Nx)
y = np.linspace(-1, 1, Ny)
u = np.zeros((Nx, Ny))

# 初始条件
for i in range(Nx):
    for j in range(Ny):
        if (x[i]**2 / a**2 + y[j]**2 / b**2 <= 1):
            u[i, j] = 1
        else:
            u[i, j] = -1

# 边界条件
```

```
u[:, 0] = u[:, -1] = u[0, :] = u[-1, :] = -1

# 迭代求解
for n in range(Nt):
    u_new = np.copy(u)
    for i in range(1, Nx - 1):
        for j in range(1, Ny - 1):
            u_xx = (u[i+1, j] - 2*u[i, j] + u[i-1, j]) / dx**2
            u_yy = (u[i, j+1] - 2*u[i, j] + u[i, j-1]) / dy**2
            u_new[i, j] = u[i, j] + dt * (u_xx + u_yy + (1 /
epsilon) * u[i, j] * (1 - u[i, j]**2))
    u = u_new

# 绘制结果
X, Y = np.meshgrid(x, y)
plt.contourf(X, Y, u.T, levels=50, cmap='RdBu')
plt.colorbar()
plt.title(f'Reaction-Diffusion Solution with ε={epsilon}, a={a}, b=
{b}')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

5. 分析结果

通过改变 ϵ, a, b 和步长 h，可以观察到解的变化。数值解可以展示反应扩散过程中的模式形成等现象。我们可以通过上述代码实验不同参数组合，来探究这些参数对解的影响。

结论

上述方法采用隐式时间离散化和中心差分空间离散化，能够无条件稳定地求解反应扩散方程。通过调整参数，可以研究不同条件下的解的行为。