are all zero along the boundary but the tangential velocity along the boundary is no longer constant zero. Implement the scheme above without using the Thom's formula to check the second order accuracy of the base scheme: use the exact solution along the boundary for the boundary values of ω, and everything else stays the same. Set $\mathrm{Re} = 1000$ and list an error and order table at $T = 0.2$ for $1/\Delta x = 8, 16, 32, 64, 128$.

(2) Consider the same exact solution above. Use the Thom's formula for this problem. Set $\mathrm{Re} = 1000$ and list an error and order table at $T = 0.2$ for $1/\Delta x = 8, 16, 32, 64, 128$.

(3) Use the Thom's formula for the Lid Driven Cavity flow. Set $\mathrm{Re} = 10000$ and compute the solution at a given time like $T = 2$ or generate an animation of the solution changing in time. See Figure 4.3 for a numerical solution. Show your result on a 200×200 mesh or a finer mesh by giving the contour plot of stream line function using the exponentially distributed contour lines and 90 contour lines for the vorticity:

```
a=min(psi(:));
b=max(psi(:));
level1=-exp(linspace(log(-a),log(0.00001),10));
level2=exp(linspace(log(0.00001),log(b)-log(10), 10));
level3=linspace(0.1*b, b, 10);
level=[level1 level2 level3];
figure;contour(x,y,psi,level);axis equal;colorbar;
% exponentially distributed contour lines for psi
figure;contour(x,y,omega,90);axis equal;colorbar
```

4.1.5 *Project: 3D Cahn-Hilliard equation in phase field models*

The Cahn-Hilliard equation is a fourth-order PDE used in the phase field modeling. A fourth order PDE is in general harder than solving a second order Poisson equation. However, when the boundary conditions are special, it can be solved by Poisson equations. For instance, the boundary value problem on the interval $[0, 1]$:

$$\frac{d^4}{dx^4}u(x) = f(x), \quad u(0) = u(1) = 0, \quad u''(0) = u''(1) = 0.$$

is equivalent to solving the Poisson equation twice:

$$\frac{d^2}{dx^2}v(x) = f(x), \quad v(0) = v(1) = 0,$$

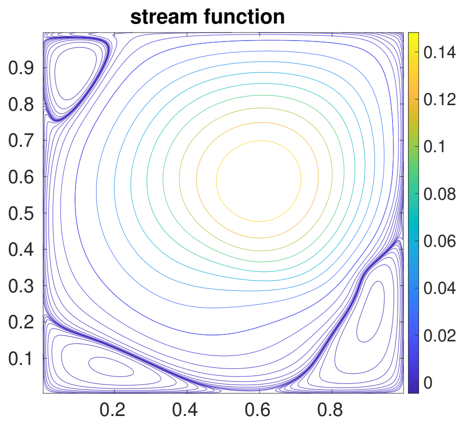

Fig. 4.3 The Lid Driven Cavity flow on a 300×300 grid for Re $= 10000$ at $T = 2$.

$$\frac{d^2}{dx^2}u(x) = v(x), \quad u(0) = u(1) = 0.$$

We consider the Cahn-Hilliard equation on $\Omega = [-1,1]^3$ with simple boundary conditions:

$$\begin{cases} \phi_t = m\Delta\left(-\epsilon\Delta\phi + \frac{1}{\epsilon}F'(\phi)\right) & \text{in } \Omega, \\ \partial_{\boldsymbol{n}}\phi = 0, \quad \partial_{\boldsymbol{n}}\Delta\phi = 0 & \text{on } \partial\Omega, \end{cases} \tag{4.7}$$

where $\phi(x,y,z,t)$ is a phase function with a thin, smooth transitional layer, whose thickness is proportional to a prescribed parameter $\epsilon > 0$, $m > 0$ is a prescribed mobility constant, and $F(\phi) = \frac{1}{4}(\phi^2-1)^2$ is a double-well potential function.

Due to the simplicity of the boundary conditions, we can avoid solving a fourth-order equation by reformulating (4.7) as a system of second-order equations after introducing the chemical potential $\mu := -\epsilon\Delta\phi + \frac{1}{\epsilon}F'(\phi)$, which is the variational derivative of the energy functional:

$$E(\phi) = \int_\Omega \frac{\epsilon}{2}|\nabla\phi|^2 + \frac{1}{\epsilon}F(\phi)d\boldsymbol{x}. \tag{4.8}$$

Then the system (4.7) is equivalent to

$$\begin{cases} \phi_t - m\Delta\mu = 0 & \text{in } \Omega, \\ \mu = -\epsilon\Delta\phi + \frac{1}{\epsilon}F'(\phi) & \text{in } \Omega, \\ \partial_{\boldsymbol{n}}\phi = 0, \quad \partial_{\boldsymbol{n}}\mu = 0 & \text{on } \partial\Omega. \end{cases} \tag{4.9}$$

For time discretization, we use the second order backward differentiation formula (BDF-2) to the system (4.9):

$$\begin{cases} \frac{\alpha\phi_{n+1}-\hat{\phi}_n}{\tau} - m\Delta\mu_{n+1} = 0, \\ \mu_{n+1} = -\epsilon\Delta\phi_{n+1} + \frac{1}{\epsilon}F'(\bar{\phi}_n), \end{cases} \tag{4.10}$$

where τ is the time step size, $\alpha = \frac{3}{2}$, $\hat{\phi}_n = 2\phi_n - \frac{1}{2}\phi_{n-1}$, and $\bar{\phi}_n = 2\phi_n - \phi_{n-1}$. To solve this linear system, we can write it as

$$\begin{pmatrix} \alpha I & -m\tau\Delta \\ \epsilon\Delta & I \end{pmatrix} \begin{pmatrix} \phi_{n+1} \\ \mu_{n+1} \end{pmatrix} = \begin{pmatrix} \hat{\phi}_n \\ \frac{1}{\epsilon}F'(\bar{\phi}_n) \end{pmatrix},$$

and its solution is given by

$$\begin{pmatrix} \phi_{n+1} \\ \mu_{n+1} \end{pmatrix} = \begin{pmatrix} \mathcal{D}I & \mathcal{D}(m\tau\Delta) \\ \mathcal{D}(-\epsilon\Delta) & \alpha\mathcal{D}I \end{pmatrix} \begin{pmatrix} \hat{\phi}_n \\ \frac{1}{\epsilon}F'(\bar{\phi}_n) \end{pmatrix} = \begin{pmatrix} \mathcal{D}(\hat{\phi}_n + \frac{m\tau}{\epsilon}\Delta F'(\bar{\phi}_n)) \\ \mathcal{D}(-\epsilon\Delta\hat{\phi}_n + \frac{\alpha}{\epsilon}F'(\bar{\phi}_n)) \end{pmatrix}, \tag{4.11}$$

where $\mathcal{D} = (\alpha I + m\tau\epsilon\Delta^2)^{-1}$.

Step I: efficient implementation with discrete Laplacian. Let Δ_h be the discrete Laplacian from second order finite difference in Chapter 2. Then the seoncd order finite difference with BDF2 for solving (4.9) can be implemented as

$$\begin{pmatrix} \phi_{n+1} \\ \mu_{n+1} \end{pmatrix} = \begin{pmatrix} \mathcal{D}_h(\hat{\phi}_n + \frac{m\tau}{\epsilon}\Delta_h F'(\bar{\phi}_n)) \\ \mathcal{D}_h(-\epsilon\Delta_h\hat{\phi}_n + \frac{\alpha}{\epsilon}F'(\bar{\phi}_n)) \end{pmatrix}, \tag{4.12}$$

where $\mathcal{D}_h = (\alpha I + m\tau\epsilon\Delta_h^2)^{-1}$. The eigenvalue decomposition $-\Delta_h = S\Lambda S^{-1}$ gives $\mathcal{D}_h = S(\alpha I + m\tau\epsilon\Lambda^2)^{-1}S^{-1}$. Thus (4.12) can be implemented in the same way as described in Section 2.9.

(1) Plug in the eigenvalue decomposition of both Δ_h and $\mathcal{D}_h$ to futher simplify (4.12).
(2) The time step can be taken as $\Delta t = \mathcal{O}(\Delta x)$. Validate the second order accuracy of this scheme on a manufactured solution

$$\phi^* = \cos(\pi x)\cos(\pi y)\cos(\pi z)\exp(t).$$

This smooth function ϕ^* no longer satisfies the original equation but a modified equation with an extra source function $g(x, y, z, t)$:

$$\phi_t = m\Delta\left(-\epsilon\Delta\phi + \frac{1}{\epsilon}F'(\phi)\right) + g,$$

where $g := \phi^*_t - m\Delta\left(-\epsilon\Delta\phi^* + \frac{1}{\epsilon}F'(\phi^*)\right)$. In other words, implement second order finite difference with BDF2 for this modified equation to validate your code by checking second order accuracy.

Step II: coalescence of two drops. Once the code is validated for its accuracy, we can use it to study the coalescence of two droplets, described by the Cahn-Hilliard equation, within the same computational domain $\Omega = [-1, 1]^3$. We select the parameter settings as $\epsilon = 0.02$, the mobility constant $m = 0.02$, and the time step size $\tau = 0.001$ with a terminal time $T = 10$. At time $t = 0$, the domain is occupied by two neighboring spherical regions of the first material, while the second material fills the remaining space, as shown in Figure 4.4. Such an initial condition is given by

$$\phi_0(\boldsymbol{x}) = 1 - \tanh\frac{|\boldsymbol{x} - \boldsymbol{x}_1| - R}{\sqrt{2}\epsilon} - \tanh\frac{|\boldsymbol{x} - \boldsymbol{x}_2| - R}{\sqrt{2}\epsilon},$$

where $\boldsymbol{x}_1 = (x_1, y_1, z_1) = (0, 0, 0.37)$ and $\boldsymbol{x}_2 = (x_2, y_2, z_2) = (0, 0, -0.37)$ are the centers of the initial spherical regions of the first material, and $R = 0.35$ is the radius of these spheres. As time progresses under the Cahn-

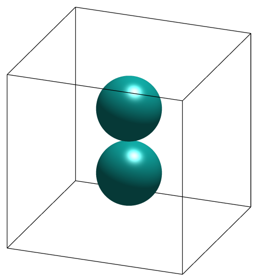

Fig. 4.4 The zero-isocontour of the phase function ϕ at time $t = 0$.

Hilliard dynamics, these two spherical regions coalesce to form a single droplet. You can also replace the Neumann boundary conditions by periodic boundary conditions to use FFT in the eigenvector method. Compare your results to the ones in [Liu *et al.* (2024)] and references therein.

Matlab code for phase function initialization

```
% Plot the initial condition for two drops
eps = 2e-2; Lx = 1;  % the domain is [-L, L]^3
R1 = 0.35*Lx; R2 = 0.35*Lx;
z1 = -0.37*Lx; z2 = 0.37*Lx;
x1 = 0; x2 = 0; y1 = 0; y2 = 0;
x = linspace(-Lx,Lx,150)'; ex = ones(size(x));
phix1 = (x - x1).^2; phiy1 = (x - y1).^2; phiz1 = (x - z1).^2;
phix2 = (x - x2).^2; phiy2 = (x - y2).^2; phiz2 = (x - z2).^2;
```

```
phi0 = 1 - tanh((sqrt(squeeze(tensorprod(phix1, ex*ex'))...
       +squeeze(tensorprod(ex,phiy1*ex'))...
       +squeeze(tensorprod(ex,ex*phiz1')))-R1)/(sqrt(2)*eps))...
       - tanh((sqrt(squeeze(tensorprod(phix2, ex*ex'))...
       +squeeze(tensorprod(ex,phiy2*ex'))...
       +squeeze(tensorprod(ex,ex*phiz2')))-R2)/(sqrt(2)*eps));
[X, Y, Z] = meshgrid(x,x,x);
fig = figure;
isosurface(X,Y,Z,phi0,0),view(-68.718,22.6096); hold on;
plotEdges(Lx,2*Lx); hold off;
set(gca,'xtick',[],'ytick',[],'ztick',[]); axis off;
```

```
function plotEdges(L,HH)
A=[ L, HH-0.5*HH, L]; B=[-L, HH-0.5*HH, L];
C=[-L,0-0.5*HH, L]; D=[ L,0-0.5*HH, L];
E=[ L, HH-0.5*HH,-L]; F=[-L, HH-0.5*HH,-L];
G=[-L,0-0.5*HH,-L]; H=[ L,0-0.5*HH,-L];
color=[0 0 0];
ptx=[A(1),B(1),C(1),D(1),A(1)];
pty=[A(2),B(2),C(2),D(2),A(2)];
ptz=[A(3),B(3),C(3),D(3),A(3)];
plot3(ptx,pty,ptz,LineWidth=0.5,Color=color); hold on
ptx=[E(1),F(1),G(1),H(1),E(1)];
pty=[E(2),F(2),G(2),H(2),E(2)];
ptz=[E(3),F(3),G(3),H(3),E(3)];
plot3(ptx,pty,ptz,LineWidth=0.5,Color=color);
plot3([A(1) E(1)],[A(2) E(2)],[A(3) ...
    E(3)],LineWidth=0.5,Color=color)
plot3([B(1) F(1)],[B(2) F(2)],[B(3) ...
    F(3)],LineWidth=0.5,Color=color)
plot3([C(1) G(1)],[C(2) G(2)],[C(3) ...
    G(3)],LineWidth=0.5,Color=color)
plot3([D(1) H(1)],[D(2) H(2)],[D(3) ...
    H(3)],LineWidth=0.5,Color=color)
axis([-L*1.2, L*1.2, -HH*0.6, HH*0.6,-L*1.2, L*1.2]); axis on;
set(gca,'FontSize',12); pbaspect([1 1 1]); hold off;
end
```

4.2 Total Variation norm for image processing

The TV norm minimization for image denoising has been proposed since early 1990s [Rudin *et al.* (1992); Sauer and Bouman (1992)]. We will first see how the model is defined, then discuss why Poisson equation is involved in popular optimization algorithms like ADMM (alternating direction method of multipliers) and Douglas-Rachford splitting for such a model. Finally, an explicit formula will be given so that one can implement