沈浙敏

双流体方程组

$$\frac{\partial n_\alpha}{\partial t}+\nabla\cdot(n_\alpha\vec{v}_\alpha)=0$$

$$m_\alpha n_\alpha\frac{d\vec{v}_\alpha}{dt}=-\nabla p_\alpha+n_\alpha\vec{F}_\alpha+\vec{R}_\alpha$$

$$\frac{3}{2}n_\alpha\frac{dT}{dt}=-p_\alpha\nabla\cdot\vec{v}_\alpha+Q_\alpha$$

$$\nabla\cdot\vec{E}_1=\frac{\sum q_\alpha n_\alpha}{\epsilon_0}$$

$$\nabla\cdot\vec{B}_1=0$$

$$\nabla\times\vec{E}_1=-\frac{\partial\vec{B}_1}{\partial t}$$

$$\nabla\times\vec{B}_1=\mu_0\sum n_\alpha q_\alpha\vec{v}_\alpha$$

摩擦项$\vec{R}_\alpha$写成Braginski形式:

$$\vec{R}_e=-\nu_{ei}n_e m_e(\vec{v}_e-\vec{v}_i)=-\eta e^2n^2(\vec{v}_e-\vec{v}_i)=\eta en\vec{j}$$

碰撞造成的能量交换Q_α:

$$Q_e=\frac{3e^2n_e^2}{m_i}(T_i-T_e)\eta$$

归一化

以$x=x_0x'$，$\vec{B}=B_0\vec{B}'$，$n=n_0n'$，$q=eq'$，$m=m_0m'$，$v=v_0v'$为基础，表示其他归一化物理量，其中$v_0=v_A=B0/\sqrt{\mu_0\rho_0}$

$$t_0=\frac{x_0}{v_0}=\frac{x_0\sqrt{\mu_0\rho_0}}{B_0}$$

$$E_0=v_0B_0=\frac{B_0^2}{\sqrt{\mu_0\rho_0}}$$

$$j_0=\frac{B_0}{\mu_0x_0}$$

$$\eta_0=\frac{E_0}{j_0}=x_0B_0\sqrt{\frac{\mu_0}{\rho_0}}$$

$$p_0=en_0E_0x_0=en_0x_0\frac{B_0^2}{\sqrt{\mu_0\rho_0}}$$

$$T_0=\frac{p_0}{n_0}=ex_0\frac{B_0^2}{\sqrt{\mu_0\rho_0}}$$

归一化后的方程为

$$\frac{\partial n_\alpha}{\partial t} + \nabla \cdot (n_\alpha \vec{v}_\alpha) = 0$$

$$m_\alpha n_\alpha \frac{d\vec{v}_\alpha}{dt} = \frac{e x_0 \sqrt{\mu_0 \rho_0}}{m_0} [-\nabla p_\alpha + n_\alpha \vec{F}_\alpha + \vec{R}_\alpha]$$

$$\frac{3}{2} n_\alpha \frac{dT}{dt} = -p_\alpha \nabla \cdot \vec{v}_\alpha + \frac{\mu_0 e^2 n_0 x_0^2}{m_0} Q_\alpha$$

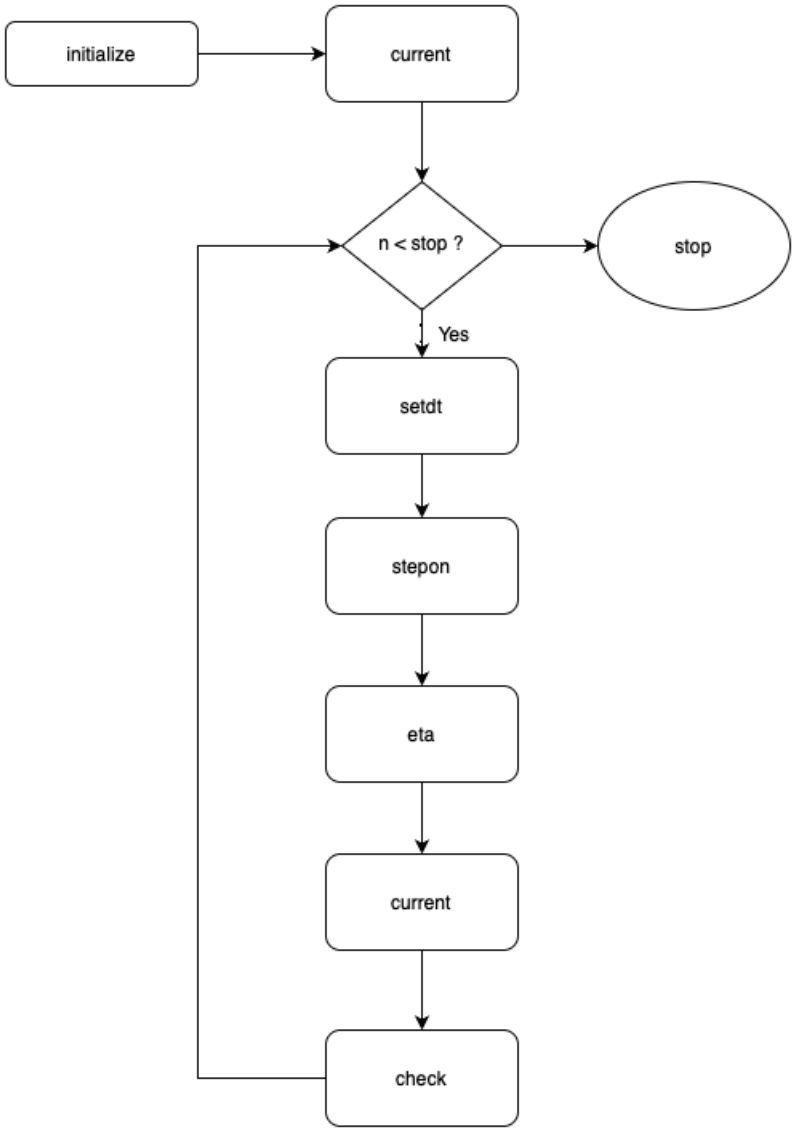
$$\nabla \cdot \vec{E}_1 = \frac{e n_0 x_0 \sqrt{\mu_0 \rho_0}}{B_0^2} \frac{\sum q_\alpha n_\alpha}{\epsilon_0}$$

$$\nabla \cdot \vec{B}_1 = 0$$

$$\nabla \times \vec{E}_1 = -\frac{\partial \vec{B}_1}{\partial t}$$

$$\nabla \times \vec{B}_1 = \mu_0 \sum n_\alpha q_\alpha \vec{v}_\alpha$$

程序流程图



子程序

```
subroutine initialize
        ! set and initialize variables
        implicit none
        integer :: i

        ! grid
        xmax = 10
        ymax = 10
        zmax = 10
        xmin = -xmax
        ymin = -ymax
        zmin = -zmax
        Nx = 101
        Ny = 101
        Nz = 101

        ! time steps
        n = 0
        nmax = 10000
        time_step_file = 40.0
        nmax_file = 20

        ! density
        ne = 1
        ni = 1

        ! velocity
        vex = 0
        vey = 0
        vez = 0
        vix = 0
        viy = 0
        viz = 0

        ! temperature
        Te = 1
        Ti = 1

        ! eta
        eta = 0.0
        is_foreta = .false.

        ! R_alpha
        Rex = 0
        Rey = 0
        Rez = 0
        Rix = -Rex
```

```
        Riy = -Rey
        Riz = -Rez
        is_R = .true.

        ! Q_alpha
        Qe = 0
        Qi = -Qe
        is_Q = .true.

        ! self-consistent EM field
        Bsx = 0
        Bsy = 0
        Bsz = 0
        Esx = 0
        Esy = 0
        Esz = 0

        ! external EM field
        Bex = 0
        Bey = 0
        Bez = 0
        Eex = 0
        Eey = 0
        Eez = 0

        ! derived parameter
        dx = (xmax-xmin)/(Nx-1.0)
        dy = (ymax-ymin)/(Ny-1.0)
        dz = (zmax-zmin)/(Nz-1.0)
        xgrid = [(xmin+i*dx, i = 0, Nx-1)]
        ygrid = [(ymin+i*dy, i = 0, Ny-1)]
        zgrid = [(zmin+i*dz, i = 0, Nz-1)]
end subroutine
```

```
subroutine total_EM_field
        implicit none
        Bx = Bsx + Bex
        By = Bsy + Bey
        Bz = Bsz + Bez
        Ex = Esx + Eex
        Ey = Esy + Eey
        Ez = Esz + Eez
end subroutine
```

```
subroutine current
        implicit none
        !借用Hall MHD代码
end subroutine
```

```
subroutine setdt
        implicit none
        !借用Hall MHD代码
end subroutine
```

```
subroutine stepon
        implicit none
        !采用两步Lax-Wendroff格式，主体借用Hall MHD代码
end subroutine
```

```
subroutine foreta
        implicit none
        if(.not. is_foreta) return
        !借用Hall MHD代码
end subroutine
```

```
subroutine check
        ! check if the code is in good running
        implicit none
        real(kind=8) :: divB = 0.0d0, temp = 0.0d0
        integer :: i, j, k

        do k = 1, Nz-1
                do j = 1, Ny-1
                        do i = 1, Nx-1
                                temp = (Bs(i+1,j,k)-Bs(i,j,k))/dx+(Bs(i,j+1,k)-
Bs(i,j,k))/dy+(Bs(i,j,k+1)-Bs(i,j,k))/dz
                                divB = max(temp, divB)
                        end do
                end do
        end do

        print *, "divB = ", divB
end subroutine
```

```
subroutine R_alpha
        implicit none

        if(.not. is_R) return
        Rex = eta*e*ne*jx
        Rey = eta*e*ne*jy
        Rey = eta*e*ne*jz
        Rix = -Rex
        Riy = -Rey
        Riz = -Rez
end subroutine
```

```
subroutine Q_alpha
        implicit none

        if(.not. is_Q) return
        Qe = 3 * e**2 * ne**2 * eta * (Ti-Te) / mi
        Qi = -Qe
end subroutine
```