OCTA

Integrated simulation system for soft materials

Multi-Phase Dynamics Program

Muffin

version 4.1

User's Manual

- Volume III -

Electrolyte Fluid Dynamics Simulator

Electrolyte

OCTA User's Group

March 03 2005

Authors of the Manual

Makoto Sasaki and Takashi Taniguchi

Programers

Takashi Taniguchi (FDM) and Makoto Sasaki (FEM)

Version 4.1 release

Programer, Authors of the Manual Tatsuya Yamaue

Acknowledgment

This work is supported by the national project, which has been entrusted to the Japan Chemical Innovation Institute (JCII) by the New Energy and Industrial Technology Development Organization (NEDO) under METI's Program for the Scientific Technology Development for Industries that Creates New Industries.

Contents

List of Figures

Chapter 1

Theoretical background of Electrolyte

"MUFFIN-Electrolyte" simulators are designed for electrolyte systems. Basic theory of these simulators is similar to that for the fluid system phase separation simulator "MUFFIN-PhaseSeparation" described in **??**. Here, we describe field parameters and dimensionless equations specific to these simulators.

1.1 Basic equations of Electrolyte

1.1.1 Notation of parameters and definition of variables

Notation of symbols for Electrolyte simulator is as follows.

e	Unit of charge ($= -1.602 \times 10^{-19}$ C(oulomb))
ϵ_o	Dielectric constant of vacuum ($=8.854 \times 10^{-12}$ $\mathrm{C^2N^{-1}m^{-2}}$)
ϵ_r	Relative dielectric constant of the pure water($=78.2$)
η_w	Viscosity of water ($=0.89 \times 10^{-3}$ Pa $\cdot$ sec $=0.89 \times 10^{-2}$ Poise)
N_c	Total number of ion components in the electrolyte solution
Z_α	Valence of the α-ion
k_BT	Product of the Boltzmann constant k_B and temperature ($1k_BT = 4.12 \times 10^{-21}$ J at $T = 298K(25^oC)$)
D_α	Diffusion constant of the α-ion.

$C_\alpha(\boldsymbol{r})$	Concentration field of the α-ion
$\boldsymbol{v}(\boldsymbol{r})$	Velocity field
$p(\boldsymbol{r})$	Pressure field
$\Phi(\boldsymbol{r})$	Scalar electric potential
$\rho_e(\boldsymbol{r})$	Charge density
$\boldsymbol{J}_\alpha(\boldsymbol{r})$	Ion flux of the α-ion
$\boldsymbol{K}(\boldsymbol{r})$	Source term in the Stokes equation

Here the subscript α denotes a component of ions, and runs from 0 to $N_c - 1$.

1.1.2 Equations of time evolution

The equation of concentration C_α (number of ions per unit volume) is described as

$$\frac{\partial C_\alpha}{\partial t} = -\nabla \cdot (\boldsymbol{v} C_\alpha) - \nabla \cdot \boldsymbol{J}_\alpha. \tag{1.1}$$

In the equation above, $\boldsymbol{J}_\alpha$ is the flux density of each ion,

$$\boldsymbol{J}_\alpha = C_\alpha(\boldsymbol{v}_\alpha - \boldsymbol{v}) \tag{1.2}$$

here, $\boldsymbol{v}_\alpha$ is the velocity of the α-ion.

When we assume a balance between the friction force and the force caused by the spatial gradient of electro-chemical potential,

$$-\xi_\alpha(\boldsymbol{v}_\alpha - \boldsymbol{v}) - \nabla\mu_\alpha = 0, \tag{1.3}$$

$$\mu_\alpha \equiv k_B T \log C_\alpha + e\mathsf{Z}_\alpha\Phi, \tag{1.4}$$

where ξ_α is a friction constant between the α-ion and the solvent. We can get the following expression,

$$\begin{aligned} \boldsymbol{J}_\alpha &= -\frac{C_\alpha}{\xi_\alpha}\nabla\mu_\alpha \equiv L_\alpha \boldsymbol{K}_\alpha, \\ \boldsymbol{K}_\alpha &\equiv -C_\alpha\nabla\mu_\alpha \\ &= -(k_B T\nabla C_\alpha + e\mathsf{Z}_\alpha C_\alpha\nabla\Phi), \end{aligned} \tag{1.5}$$

where $L_a \equiv 1/\xi_\alpha$ is the Onsager transport coefficient. The equation of the velocity field is written as

$$-\nabla p + \eta_w\nabla^2\boldsymbol{v} + \boldsymbol{K} = 0, \tag{1.6}$$

where $\boldsymbol{K}$ is the source term for Stokes equation and calculated as the body force caused by a friction between the ion flux and the solvent flow,

$$\begin{aligned} \boldsymbol{K} &= \sum_{\alpha=0}^{N_c-1} C_\alpha\xi_\alpha(\boldsymbol{v}_\alpha - \boldsymbol{v}) = -\sum_{\alpha=0}^{N_c-1} C_\alpha\nabla\mu_\alpha = \sum_{\alpha=0}^{N_c-1}\boldsymbol{K}_\alpha \\ &= -\sum_{\alpha=0}^{N_c-1}\left[k_B T\nabla C_\alpha + e\mathsf{Z}_\alpha C_\alpha\nabla\Phi\right]. \end{aligned} \tag{1.7}$$

The electric potential obeys the Poisson equation:

$$\nabla^2\Phi = -\frac{1}{\epsilon_o\epsilon_r}\sum_\alpha e\mathsf{Z}_\alpha C_\alpha. \tag{1.8}$$

1.2 Units

In this section, we describe units for our electrolyte simulator and dimensionless expressions. We will use the MKSA unit system throughout this section.

1.2.1 Units for length and time

As the unit of length, we adopt a length l which can be expressed by the Bjerrum length ξ_B

$$l = 4\pi\xi_B = \frac{1}{\epsilon_o\epsilon_r}\frac{e^2}{k_B T}. \tag{1.9}$$

The unit of time is determined using the typical value of diffusion time of an ion such as $\mathrm{Na}^{(+)}$ in water. We use $D^* = 1.0\times10^{-5}$ cm^2/sec as the typical value of diffusion constant. Using the characteristic length l, a characteristic time τ is given by

$$\tau = l^2/D^*. \tag{1.10}$$

In the case of a typical electrolyte solution which contains an ion having a diffusion constant D^* at $T = 298$K (25^oC), the units of length l and time τ can be estimated to be

$$l = 8.96 \text{ nm}, \tag{1.11}$$

$$\tau = 0.803\times10^{-7} \text{ sec} = 0.08\ \mu\text{sec}. \tag{1.12}$$

A typical magnitude of velocity derived from the above two units is

$$v = \frac{l}{\tau} = 1.116\times10^{-1}\text{m/sec}. \tag{1.13}$$

1.2.2 Dimensionless expression for time evolution equations

In the followings, variables with a tilde mean dimensionless ones. We adopt l and τ as the units of space and time, so that $\tilde{x}_i \equiv x_i/l$ $(i = 1, 2, 3)$ and $\tilde{t} \equiv t/\tau$. The concentration C_α and the velocity $\boldsymbol{v}$ are scaled by $C^* = 10^{-3}$ mol/liter $= 6.023 \times 10^{23}$ ions/m^3 and $v^*(\equiv l/\tau)$, respectively. The diffusion constant $D_\alpha = k_B T L_\alpha$ is scaled by the typical value D^*, and then $\tilde{D}_\alpha = D_\alpha/D^*$. Then, the dimensionless equation for the concentration of the α-ion is

$$\frac{\partial \tilde{C}_\alpha}{\partial \tilde{t}} = -\tilde{\nabla} \cdot (\tilde{\boldsymbol{v}} \tilde{C}_\alpha) - \tilde{\nabla} \cdot \tilde{\boldsymbol{J}}_\alpha. \tag{1.14}$$

In the above equation, $\tilde{\boldsymbol{J}}_\alpha$ is defined as

$$\tilde{\boldsymbol{J}}_\alpha \equiv \tilde{\mathsf{D}}_\alpha \tilde{\boldsymbol{K}}_\alpha, \tag{1.15}$$

$$\tilde{\boldsymbol{K}}_\alpha \equiv -\left[\tilde{\nabla}\tilde{C}_\alpha + \tilde{\mathsf{R}}\ \mathsf{Z}_\alpha \tilde{C}_\alpha \tilde{\nabla}\tilde{\Phi}\right], \tag{1.16}$$

where a dimensionless parameter $\tilde{R}$ is introduced. The parameter $\tilde{\mathsf{R}}$ is defined as

$$\tilde{\mathsf{R}} = \frac{e\Phi_o}{k_B T}. \tag{1.17}$$

The electric potential obeys the Poisson equation given by

$$\tilde{\nabla}^2 \tilde{\Phi} = -\frac{\tilde{\mathsf{M}}}{\tilde{\mathsf{R}}} \sum_\alpha \mathsf{Z}_\alpha \tilde{C}_\alpha, \tag{1.18}$$

where $\tilde{\mathsf{M}} \equiv C^* l^3$ (mol). The equation of the velocity field is written as

$$-\tilde{\nabla}\tilde{p} + \tilde{\nabla}^2 \tilde{\boldsymbol{v}} + \tilde{\mathsf{M}}\tilde{\mathsf{D}}\ \tilde{\boldsymbol{K}} = 0, \tag{1.19}$$

where

$$\tilde{\mathsf{D}} = \mathsf{D}^{(l)}/\mathsf{D}^* \qquad \text{and} \qquad \mathsf{D}^{(l)} = \frac{k_B T}{6\pi\eta_w (l/6\pi)}, \tag{1.20}$$

and dimensionless volume force $\tilde{\boldsymbol{K}}$ is

$$\tilde{\boldsymbol{K}} = \sum_\alpha \tilde{\boldsymbol{K}}_\alpha = -\sum_\alpha \left[\tilde{\nabla}\tilde{C}_\alpha + \tilde{\mathsf{R}}\ \mathsf{Z}_\alpha \tilde{C}_\alpha \tilde{\nabla}\tilde{\Phi}\right]. \tag{1.21}$$

Hereafter, we will omit the tilde for dimensionless variables to simplify expressions. The set of equations becomes as follows

$$\frac{\partial C_\alpha}{\partial t} = -\nabla \cdot (\boldsymbol{v} C_\alpha) - \nabla \cdot \boldsymbol{J}_\alpha, \tag{1.22}$$

$$\boldsymbol{J}_\alpha \equiv -\mathsf{D}_\alpha \left[\nabla C_\alpha - \mathsf{R}\ \mathsf{Z}_\alpha C_\alpha \nabla\Phi\right], \tag{1.23}$$

$$\nabla^2 \Phi = -\frac{\mathsf{M}}{\mathsf{R}} \sum_\alpha \mathsf{Z}_\alpha C_\alpha, \tag{1.24}$$

$$-\nabla p + \nabla^2 \boldsymbol{v} + \mathsf{M}\mathsf{D}\ \boldsymbol{K} = 0. \tag{1.25}$$

The parameters in the above equations are

$$\mathsf{D}_\alpha, \qquad \mathsf{Z}_\alpha, \qquad \mathsf{R}, \qquad \mathsf{M} \quad \text{and} \quad \mathsf{D}. \tag{1.26}$$

In the case of T= 298K, the unit length is $l = 8.96 \times 10^{-9}$m $= 8.96$ nm. Using this unit length l, $M = C^* l^3 = 4.33 \times 10^{-1}$ (ions) and $\mathsf{D}_\mathsf{l} = 5.17 \times 10^{-6}$ cm^2/sec. In the above estimation, the unit of the electric potential $\Phi_o = 1$mV is used and we finally obtain the values of following parameters.

$$\mathsf{R} = 3.89 \times 10^{-2}, \qquad \mathsf{M} = 4.33 \times 10^{-1} \quad \text{and} \quad \mathsf{D} = 0.517. \tag{1.27}$$

1.3 The Poisson-Boltzmann equation and the Debye Length

In the equilibrium state of an electrolyte system, the electro-chemical potential should be homogeneous in space;

$$\begin{aligned} \nabla\mu_\alpha &= \nabla\left[k_BT\log C_\alpha + e\mathsf{Z}_\alpha\Phi\right] \\ &= k_BT\frac{\nabla C_\alpha}{C_\alpha} + e\mathsf{Z}_\alpha\nabla\Phi = 0. \end{aligned} \tag{1.28}$$

With boundary conditions $\Phi|_{\boldsymbol{r}=\infty} = 0$ and $C_\alpha|_{\boldsymbol{r}=\infty} = C_\alpha^\infty$, the equation gives

$$C_\alpha(\boldsymbol{r}) = C_\alpha^\infty \exp\left(-\frac{e\mathsf{Z}_\alpha\Phi(\boldsymbol{r})}{k_BT}\right). \tag{1.29}$$

Substituting this equation to (1.8), we get an equation known as the Poisson-Boltzmann equation:

$$\nabla^2\Phi = -\frac{1}{\epsilon_o\epsilon_r}\sum_\alpha e\mathsf{Z}_\alpha C_\alpha^\infty \exp\left(-\frac{e\mathsf{Z}_\alpha\Phi(\boldsymbol{r})}{k_BT}\right). \tag{1.30}$$

Applying an approximation $e\mathsf{Z}_\alpha\Phi/k_BT \ll 1$ which is known as the Debye-Hückel approximation, we obtain an equation:

$$\nabla^2\Phi = -\frac{1}{\epsilon_o\epsilon_r}\sum_\alpha e\mathsf{Z}_\alpha C_\alpha^\infty + \frac{\sum_\alpha (e\mathsf{Z}_\alpha)^2 C_\alpha^\infty}{\epsilon_o\epsilon_r k_BT}\Phi. \tag{1.31}$$

The first term of the right hand side disappears because of the charge neutrality requirement. For one dimensional system we get

$$\frac{d^2\Phi(x)}{dx^2} = \kappa^2\Phi, \tag{1.32}$$

$$\frac{1}{\kappa} \equiv \sqrt{\frac{\epsilon_o\epsilon_r k_BT}{\sum_\alpha (e\mathsf{Z}_\alpha)^2 C_\alpha^\infty}}. \tag{1.33}$$

The parameter $1/\kappa$ with a dimension of length is called "Debye length". The electric potential has the form of $\Phi(x) = \Phi_0 exp(-\kappa x)$ in this approximation, and it is explained that an electric field by electric charge is "shielded" by counter ions within the order of the Debye length.

According to the dimensionless expressions considered in 1.2.2, the Debye length in this unit system is calculated as

$$\begin{aligned} & \tilde\nabla\tilde C_\alpha + \tilde{\mathsf{R}}\,\mathsf{Z}_\alpha\tilde C_\alpha\tilde\nabla\tilde\Phi = 0 \\ \rightarrow\quad & \tilde C_\alpha = \tilde C_\alpha^\infty \exp\left(-\tilde{\mathsf{R}}\mathsf{Z}_\alpha\tilde\Phi\right) \\ \rightarrow\quad & \tilde\nabla^2\Phi = -\frac{\tilde{\mathsf{M}}}{\tilde{\mathsf{R}}}\sum_\alpha \mathsf{Z}_\alpha\tilde C_\alpha^\infty \exp\left(-\tilde{\mathsf{R}}\mathsf{Z}_\alpha\tilde\Phi\right) \approx \left(\tilde{\mathsf{M}}\sum_\alpha \mathsf{Z}_\alpha^2\tilde C_\alpha^\infty\right)\tilde\Phi. \end{aligned}$$

Thus, we get a dimensionless "Debye length" as follows.

$$\frac{1}{\tilde\kappa} \equiv \sqrt{\frac{1}{\tilde{\mathsf{M}}\sum_{\alpha=0}^{N_c-1}\mathsf{Z}_\alpha^2\tilde C_\alpha^\infty}}. \tag{1.34}$$

Using $\tilde{\mathsf{M}} = 4.33\times10^{-1}$, $\tilde C_\alpha^\infty = 1.0, \mathsf{Z}_\alpha = \pm1$ and $N_c = 2$, we get $\frac{1}{\tilde\kappa} = 1.075\times l$.

1.4 Boundary condition on the surface of particles

1.4.1 Boundary condition for the electric potential

The boundary condition for the electric potential Φ at the surface of a particle having a surface charge density σ can be expressed by the following Neumann boundary condition

$$(\boldsymbol{n} \cdot \nabla\Phi)|_S = -\sigma/\epsilon_r\epsilon_o \tag{1.35}$$

where the subscript $|_S$ means the equation evaluates at the Surface of a particle, and $\boldsymbol{n}$ denotes a unit vector normal to the surface. Using the units of length l and the scalar electric potential Φ_o, the boundary condition (1.35) is written as

$$(\boldsymbol{n} \cdot \nabla\Phi)|_S = -\frac{q}{\mathsf{R}}, \tag{1.36}$$

$$q \equiv \frac{\sigma l^2}{e}. \tag{1.37}$$

Here q denotes the effective number of dissociation functions having valence $Z = 1$.

1.4.2 Boundary condition for the velocity and pressure

The boundary conditions of the velocity field, pressure and ion concentration on the surface are expressed as

$$\boldsymbol{v}|_S = 0 \tag{1.38}$$

$$(\boldsymbol{n} \cdot \nabla P)|_S = (\boldsymbol{n} \cdot \nabla^2\boldsymbol{v})|_S \tag{1.39}$$

$$(\boldsymbol{n} \cdot \boldsymbol{K}_\alpha)|_S = 0. \tag{1.40}$$

1.4.3 Actual and dimensionless parameters required for input

To get a value of a dimensionless parameter, at first you have to input the following basic parameters (actual value with a unit).

[basic parameters (actual value with a unit)]

Notation	Meanings
N_c	number of components
C_α	Concentration of each ion component ($\alpha = 0, \cdots N_c - 1$)
D_α	diffusion coefficient of each component
Z_α	valence of each ion component
T	temperature

Space unit ξ, time unit τ, etc. can be derived from these input.

[space unit and time unit]

Meanings	Notation	Expression
space unit	l	$l = 4\pi\xi_B = \frac{1}{\epsilon_0\epsilon_r}\frac{e^2}{k_BT}$
time unit	τ	$\tau = l^2/D^*$

[Conversion from an actual to a dimensionless parameter]

Dimensionless parameter	meanings	expression for parameter	input parameter name in MUFFIN
$\tilde{R}$	ratio of electrostatic energy and thermal energy	$R = e\Phi_0/k_BT$	`R`
$\tilde{D}$	see (1.2.2)		`D`
$\tilde{M}$	see (1.2.2)		`M`
$\tilde{D}_\alpha$	dimensionless diffusion coefficient	$D = D_\alpha/D^*$	`DIFFUSION_COEFFICIENT`
$\tilde{\sigma}$	Charge density on particle surface	$q = \sigma l^2/e$	`SURFACE_CHARGE_ON_OBSTACLE`

1.5 FDM electrolyte simulator Electrolyte_FDM

Electrolyte_FDM is an electrolyte simulator of multicomponent ion systems using the finite difference method.

List of selectable fields

Selectable fields	notation
Ion concentration field	C_α
Flux field (without hydrodynamics effect)	$\boldsymbol{J}_\alpha$
Velocity field	$V_i \quad (i = x, y \ or \ z)$
Pressure field	P
Electric potential field	Φ

The greek index α expresses a component which takes a value of $\alpha = 0, \cdots N_c - 1$ (N_c: the number of components).

1.5.1 Boundary condition on surface of "Obstacle" in electrolyte solution

In the FDM electrolyte simulator, we can use "obstacle" with a surface charge in an electrolyte solution, and the distribution of the ion and a flow field around it can be calculated. The obstacle is expressed as a field named "Obstacle". Boundary conditions of each field on the surface of "Obstacle" is explained below.

Boundary condition of the electrostatic potential on obstacle

The boundary condition of the electrostatic potential on the surface of Obstacle with a surface charge density σ is described by the following Neumann boundary condition,

$$(\boldsymbol{n} \cdot \nabla\Phi)|_S = -\sigma/\epsilon_r\epsilon_o. \tag{1.41}$$

Subscript $|_S$ means the value on the surface of Obstacle. Using the units of length l and electrostatic potential Φ_o, boundary condition (1.41) is given in a dimensionless form as

$$(\boldsymbol{n} \cdot \nabla\Phi)|_S = -\frac{q}{\mathsf{R}}, \tag{1.42}$$

$$q \equiv \frac{\sigma l^2}{e}. \tag{1.43}$$

Boundary conditions of the velocity, pressure, and concentration field on the surface of obstacle

The boundary conditions on the surface of Obstacle for velocity, pressure, and concentration fields are as follows:

$$\boldsymbol{v}|_S = 0, \tag{1.44}$$

$$(\boldsymbol{n} \cdot \nabla P)|_S = (\boldsymbol{n} \cdot \Delta\boldsymbol{v})|_S, \tag{1.45}$$

$$(\boldsymbol{n} \cdot \boldsymbol{J}_\alpha)|_S = 0. \tag{1.46}$$

In the Electrolyte_FDM simulator, these boundary conditions will be set up automatically when Obstacle exists.

1.5.2 The boundary conditions on the wall surfaces for each field

In the electrolyte simulator using the finite difference method, the shape of a system is rectangular. Therefore, each field must have boundary conditions for the six boundaries shown in Figure **??**.

Boundary conditions of ion concentration field C_α

Possible boundary conditions for the ion concentration field are as follows.

- **Periodic boundary condition**
  When a periodic boundary condition is applied in x-direction,

  $$C_\alpha(x, y, z) = C_\alpha(x + L_x, y, z).$$

  The similar condition can be applied for y- and z-directions.

- **Biased periodic boundary condition**
  A biased periodic condition is similar to the periodic boundary condition, but at two facing boundaries the value of the field may have a non-zero gap. When this condition is applied for x-direction,

  $$C_\alpha(x, y, z) = C_\alpha(x + L_x, y, z) + A_x,$$

  where A_x is a gap for x-direction.

- **Wall boundary condition**
  At each of six boundaries of rectangular geometry, a gradient of the field along the direction perpendicular to the boundary can be set to zero:

  $$\boldsymbol{n} \cdot \nabla C_\alpha(x, y, z)|_{wall} = 0,$$

  where $|_{wall}$ denotes the value on the wall.

- **Bulk boundary condition**

  At each of six boundaries of rectangular geometry, a constant value can be given for the field. It means that the value of concentration is constant outside of the boundary (bulk),

  $$C_\alpha(x, y, z)|_{Boundary} = \text{Constant}_\alpha.$$

Boundary conditions for flux field $\boldsymbol{K}_\alpha$

Possible boundary conditions for the flux field are as follows.

- **Periodic boundary condition**
  When a periodic boundary condition is applied in x-direction,

  $$K_{\alpha x}(x, y, z) = K_{\alpha x}(x + L_x, y, z).$$

  The similar condition can be applied for y- and z-directions.

- **Wall boundary condition**
  At each of six boundaries of rectangular geometry, flux can be set to zero. It means that flux by diffusion is zero on the boundary wall:

  $$\boldsymbol{K}_\alpha(x, y, z)|_{wall} = \boldsymbol{0},$$

  where $|_{wall}$ denotes the value on the wall.

- **Bulk boundary condition**
  At each of six boundaries of rectangular geometry, a gradient of the flux in the direction perpendicular to the boundary can be set to zero. It means that flux by diffusion is zero because of a constant value of concentration field outside the boundary (bulk):

  $$(\boldsymbol{n} \cdot \nabla)\boldsymbol{K}_\alpha(x, y, z)|_{Boundary} = \boldsymbol{0}.$$

Boundary conditions for velocity field $\boldsymbol{v}$

Possible boundary conditions for the velocity field are as follows.

- **Periodic boundary condition**
  When a periodic boundary condition is applied in x-direction,

  $$\boldsymbol{v}(x, y, z) = \boldsymbol{v}(x + L_x, y, z).$$

  The similar condition can be applied for y- and z-directions.

- **Constant velocity boundary condition**
  At each of six boundaries of rectangular geometry, velocity vector can be set to a constant vector $\boldsymbol{v}_o$. This means that the boundary wall is moving with the constant velocity:

  $$\boldsymbol{v}(x, y, z)|_{wall} = \boldsymbol{v_o},$$

  where $|_{wall}$ denotes the value on the wall, and $\boldsymbol{v}_o \perp \boldsymbol{n}$.

- **When constant pressure boundary condition is applied**
  The gradient of velocity should be set to zero on the boundary:

  $$(\boldsymbol{n} \cdot \nabla)\boldsymbol{v}(x, y, z)|_{wall} = \boldsymbol{0}.$$

Boundary conditions for pressure field P

Possible boundary conditions for the pressure field are as follows.

- **Periodic boundary condition**
  When a periodic boundary condition is applied in x-direction,

  $$P(x, y, z) = P(x + L_x, y, z).$$

  The similar condition can be applied for y- and z-directions.

- **Biased periodic boundary condition**
  A biased periodic condition is similar to the periodic boundary condition, but at two facing boundaries the value of the field may have a non-zero gap. When this condition is applied for x-direction,

  $$P(x, y, z) = P(x + L_x, y, z) + A_x,$$

  where A_x is a gap for x-direction.

- **When constant velocity boundary condition is applied**
  The gradient of pressure should be set to zero on the boundary:

  $$\boldsymbol{n} \cdot \nabla P(x, y, z)|_{Boundary} = 0.$$

- **Constant pressure boundary condition**
  At each of six boundaries of rectangular geometry, pressure can be set to a constant value P_o:

  $$P(x, y, z)|_{Boundary} = P_o.$$

- **Oscillating pressure on a boundary**
  The pressure oscillates with a constant frequency on a boundary.

  $$P(x, y, z)|_{Boundary} = P_o + \delta P \cdot \sin(\omega t),$$

  where P_o is a constant pressure, δP is a oscillation amplitude, ω is a frequency, and t is time.

Boundary conditions for electric potential field

Possible boundary conditions for the electric potential field are as follows.

- **Periodic boundary condition**
  When a periodic boundary condition is applied in x-direction,

  $$\phi(x, y, z) = \phi(x + L_x, y, z).$$

  Similar condition can be applied for y- and z-directions.

- **Biased periodic boundary condition**

  A biased periodic condition is similar to the periodic boundary condition, but at two facing boundaries the value of the field may have a non-zero gap. When this condition is applied for x-direction,

  $$\phi(x, y, z) = \phi(x + L_x, y, z) + A_x,$$

  where A_x is a gap for x-direction.

- **Surface charge density given on a boundary(Neumann boundary condition)**

  A constant surface charge on a boundary can be given by the gradient of electric potential,

  $$\boldsymbol{n} \cdot \nabla\phi(x, y, z)|_{Boundary} = -\sigma/\epsilon_r\epsilon_o.$$

  In a dimensionless form,

  $$\boldsymbol{n} \cdot \nabla\phi(x, y, z)|_{Boundary} = -q/\mathsf{R},$$

  where definition of dimensionless parameters given by equations (1.42) and (1.43).

- **Constant potential on a boundary (Dirichlet boundary condition)**
  At each of six boundaries of rectangular geometry, the electric potential can be set to a constant value ϕ_o:

  $$\phi(x, y, z)|_{Boundary} = \phi_o \qquad (= \text{Constant}).$$

- **Oscillating electric potential on a boundary**
  The electric potential oscillates with a constant frequency on a boundary:

  $$\phi(x, y, z)|_{Boundary} = \phi_o + \delta\phi \cdot \sin(\omega t),$$

  where $\phi_o$is a constant potential, δP is a oscillation amplitude, ω is a frequency, and t is time.

1.6 FEM electrolyte simulator Electrolyte_FEM

The Electrolyte_FEM is an electrolyte simulator using the finite element method. The characteristic features of this simulator are as follows.

- The finite element method of a three-dimensional Euler picture is used as the calculation method. The supported element type is tetrahedral element with a linear interpolation shape function.
- A slow viscous flow (Stokes flow) can be dealt with.
- The Dirichlet conditions, the Neumann conditions, and a periodic boundary condition can be used as a boundary condition (periodic boundary condition is only for rectangular geometry).

Compared with the electrolyte simulator using the finite difference method Electrolyte_FDM, this FEM simulator has more flexibility in geometry,and, in general, boundary condition treatment is easier. On the other hand, it is necessary to take care that the computation time and required memory may be increased as compared with the FDM simulator.

List of selectable fields

List of selectable fields

Selectable fields	notation
Ion concentration field	$C_\alpha \quad (\alpha = 0, 1, \cdots, N_c - 1)$
Charge density field	$\rho_e(\boldsymbol{r})$
Dielectric constant field	$\epsilon(\boldsymbol{r})$
Flux field (without hydrodynamics effect)	$\boldsymbol{J}_\alpha$
Velocity field	$\boldsymbol{V}$
Pressure field	P
Electric potential field	Φ

The greek index α expresses a component which takes a value of $\alpha = 0, \cdots N_c - 1$ (N_c: the number of components).

1.6.1 Calculation model

Calculation model of Electrolyte_FEM is similar to that of Electrolyte_FDM, and FEM method treatment is similar to that of PhaseSeparation_FEM (see section **??**).

1.6.2 Treatment of external electric field

In Electrolyte_FEM simulator, we prepared a special method for treatment of a constant external electric field applied over a calculated system. In such a case, the electric potential can be expressed in two terms:

$$\phi(\boldsymbol{r}) = \phi'(\boldsymbol{r}) - \boldsymbol{E}_0 \cdot \boldsymbol{r} \tag{1.47}$$

where $\boldsymbol{E}_0$ is the strength of the external electric field. No electric charge arises from the term including $\boldsymbol{E}_0$ because $\nabla^2(\boldsymbol{E}_0 \cdot \boldsymbol{r}) = 0$, so the first term in the right hand side $\phi'(\boldsymbol{r})$ includes all the effects of internal electric charge of ions and/or charges on objects in the electrolyte solution.

In the calculation procedure of Electrolyte_FEM, $\boldsymbol{E}_0$ is added to the driving force term of the fluid equation as $\boldsymbol{K} + \rho_e \boldsymbol{E}_0$ in a processing of the flux field K_Field ("GRADIENT_ELECTRO_CHEMICAL_POTENTIAL_WITH_EXTERNAL_FIELD"). And, for the electric potential, $\phi'(\boldsymbol{r})$ should be treated as $\phi(\boldsymbol{r})$ of systems without an external electric field.

One reason for using such a method is that the Electrolyte_FEM simulator does not support the "biased periodic" boundary condition which is possible in the FDM simulators, but the more important reason is that an "external" field should be constant over a calculated system and it cannot be affected by any internal phenomena in the calculated system. So separating the electric potential into two terms as above has an important physical meaning.

1.6.3 Zero-electric-current boundary condition

In some cases of electrolyte simulations, it is useful to assume that the electric current integrated over a boundary surface becomes zero. For example, to calculate the streaming potential [1] this condition is essential.

An electric current is written as follows using the dimensionless expression:

$$\boldsymbol{j} = \sum_{\alpha} Z_{\alpha} C_{\alpha} \boldsymbol{v}_{\alpha} = \sum_{\alpha} Z_{\alpha} \left(\boldsymbol{v} C_{\alpha} + \boldsymbol{J}_{\alpha} \right), \tag{1.48}$$

where $\boldsymbol{v}_{\alpha}$ is an averaged velocity of ion species α and $\boldsymbol{J}_{\alpha}$ is an ion flux vector written as

$$\boldsymbol{J}_{\alpha} \equiv -\mathsf{D}_{\alpha} \left[\nabla C_{\alpha} - \mathsf{R}\ Z_{\alpha} C_{\alpha} \nabla \Phi \right]. \tag{1.49}$$

The zero current condition is satisfied by an external electric field caused by the streaming potential, and when defining it as $\boldsymbol{E}_0 = (\nabla \phi)_S$ on the boundary, the zero current condition is calculated as

$$\begin{aligned} \int dS \boldsymbol{j} \cdot \boldsymbol{n} &= \sum_{\alpha} \int dS Z_{\alpha} C_{\alpha} \boldsymbol{v} \cdot \boldsymbol{n} \\ &\quad - \sum_{\alpha} \int dS \mathsf{D}_{\alpha} Z_{\alpha} \boldsymbol{n} \cdot \nabla C_{\alpha}, \\ &\quad - \sum_{\alpha} \int dS \mathsf{D}_{\alpha} \mathsf{R} Z_{\alpha}^2 C_{\alpha} \left[-\boldsymbol{E}_0 \cdot \boldsymbol{n} \right] \\ &= 0. \end{aligned} \tag{1.50}$$

From this equation, we get the gradient of the electric potential on the boundary as

$$\boldsymbol{n} \cdot \nabla \phi|_s = -\boldsymbol{E}_0 \cdot \boldsymbol{n} = \frac{\sum_{\alpha} \int dS Z_{\alpha} C_{\alpha} \boldsymbol{v} \cdot \boldsymbol{n} - \sum_{\alpha} \int dS \mathsf{D}_{\alpha} Z_{\alpha} \boldsymbol{n} \cdot \nabla C_{\alpha}}{\sum_{\alpha} \int dS \mathsf{D}_{\alpha} \mathsf{R} Z_{\alpha}^2 C_{\alpha}}. \tag{1.51}$$

Currently we assume that $\sum_{\alpha} \int dS \mathsf{D}_{\alpha} Z_{\alpha} \boldsymbol{n} \cdot \nabla C_{\alpha} = 0$ on the boundary for simplicity.

This condition is applied as a Neumann condition for the electric potential field ("N_ZERO_ELECTRIC_CURRENT").

1.6.4 Boundary conditions in Electrolyte_FEM

You can use the following type of boundary conditions (partial region conditions) in the Electrolyte_FEM simulator.

- Periodic boundary condition:

  This condition is possible only when an "UNSTRUCTURED_RECT" type of mesh is specified with geometrical periodic boundary. In this case periodic boundaries are treated "geometrically" for all fields, so that the fields satisfy periodic boundary condition automatically. So you need not specify periodic condition explicitly as the partial region conditions in input UDF, or, in other word, you can't use boundary conditions other than the periodic condition on geometrical periodic boundaries.

- Dirichlet condition:

  Set a constant values in time for partial region.

- Neumann condition:

  For a gradient vector of a field, or a vector field itself, a component normal to a boundary surface is given as a constant value.

  When the value of the normal component is zero for a field, you need not specify any boundary conditions for the field in input-UDF, as explained in **??**.

The FEM simulator does not support the "biased periodic condition" and the Lees Edwards boundary condition supported in the FDM simulators.

Boundary conditions of ion concentration field C_α

Possible boundary conditions for the ion concentration field are as follows.

- **Periodic boundary condition**
  This condition is applied when the mesh type is "UNSTRUCTURED_RECT" which has a periodic boundary. When a periodic boundary condition is set in X direction, the condition requires,

  $$C_\alpha(x, y, z) = C_\alpha(x + L_x, y, z).$$

  When a periodic condition is used for Y or Z direction, the similar condition is applied for each direction. The periodic boundary condition is required by geometry, so you need not, or cannot, specify any periodic boundary condition as a partial region condition in input UDF.

- **Wall boundary condition** (Neumann condition)
  This condition sets the normal component of diffusion flux $\boldsymbol{J}_\alpha$ to zero:

  $$\boldsymbol{n} \cdot \boldsymbol{J}_\alpha(x, y, z)|_{wall} = 0.$$

  It means also;

  $$\boldsymbol{n} \cdot \nabla\psi_\alpha(x, y, z)|_{wall} = 0,$$

  where $|_{wall}$ indicates a field value on the wall. When the boundary is not a geometrical periodic boundary, and if no boundary condition is given for ψ_α, this wall condition is applied as default.

- **Bulk boundary condition** (Dirichlet condition)
  This condition means that the value of the ion concentration is constant outside of the boundary (bulk), and a constant value is given for the field on a boundary surface:

  $$C_\alpha(x, y, z)|_{Boundary} = \text{Constant}_\alpha.$$

Boundary conditions for flux field $\boldsymbol{K}_\alpha$

Possible boundary conditions for the flux field are as follows.

- **Periodic boundary condition**
  This condition is applied when the mesh type is "UNSTRUCTURED_RECT" which has a periodic boundary. When a periodic boundary is set in the X direction, the condition requires,

  $$\boldsymbol{K}_\alpha(x, y, z) = \boldsymbol{K}_\alpha(x + L_x, y, z).$$

  When a periodic condition is used for Y- or Z-direction, the similar condition is applied for each direction. The periodic boundary condition is required by geometry, so you need not, or cannot, specify any periodic boundary condition as a partial region condition in input UDF.

- **Wall boundary condition**
  This condition sets the normal component of flux $\boldsymbol{J}_\alpha$ to be zero so as not to make the fluid flow through a boundary surface (wall).

  $$\boldsymbol{n} \cdot \boldsymbol{J}_\alpha(x, y, z)|_{wall} = 0,$$

  where $|_{wall}$ indicates a field value on the wall. This condition is identical to the wall boundary condition for ψ_α. When the boundary is not a geometrical periodic boundary, and if no boundary condition is given for ψ_α, this wall condition is applied as default.

Boundary conditions for velocity field $\boldsymbol{v}$

Possible boundary conditions for the velocity field are as follows.

- **Periodic boundary condition**
  This condition is applied when the mesh type is "UNSTRUCTURED_RECT" which has a periodic boundary. When a periodic boundary is set in X direction, the condition requires,

  $$\boldsymbol{v}(x, y, z) = \boldsymbol{v}(x + L_x, y, z).$$

  When a periodic condition is used for Y- or Z-direction, the similar condition is applied for each direction. The periodic boundary condition is required by geometry, so you need not, or cannot, specify any periodic boundary condition as a partial region condition in input UDF.

- **Constant velocity boundary condition**
  On a boundary (or generally in a partial region), the velocity can be set to a constant value $\boldsymbol{v_o}$:

  $$\boldsymbol{v}(x, y, z)|_{wall} = \boldsymbol{v_o},$$

  where $|_{wall}$ denotes a value on the wall. It can be used for a case that the boundary wall is moving with a constant velocity, or not moving if set to zero. It can also be used to make "obstacle" in flow by setting a constant velocity on an internal partial region as is described in **??**.

Boundary conditions for pressure field P

Possible boundary conditions for the pressure field are as follows.

- **Periodic boundary condition**
  This condition is applied when the mesh type is "UNSTRUCTURED_RECT" which has a periodic boundary. When a periodic boundary is set in the X direction, the condition requires;

  $$P(x, y, z) = P(x + L_x, y, z).$$

  When a periodic condition is used for Y or Z direction, the similar condition is applied for each direction. The periodic boundary condition is required by geometry, so you need not, or cannot, specify any periodic boundary condition as a partial region condition in input UDF.

- **Constant pressure boundary condition**(Dirichlet condition)
  On a boundary surface (or generally in a partial region), the pressure can be set to a constant value P_o:

  $$P(x, y, z)|_{Boundary} = P_o.$$

Boundary conditions for electric potential field

Possible boundary conditions for the electric potential field are as follows.

- **Periodic boundary condition**
  This condition is applied when the mesh type is "UNSTRUCTURED_RECT" and it has a periodic boundary. When a periodic boundary is set in X direction, the condition requires;

  $$\phi(x, y, z) = \phi(x + L_x, y, z).$$

  When a periodic condition is used for Y or Z direction, the similar condition is applied for each direction. The periodic boundary condition is required by geometry, so you need not, or cannot, specify any periodic boundary condition as a partial region condition in input UDF.

- **A surface charge density is given on a boundary (Neumann boundary condition)**
  A constant surface charge on a boundary can be given by gradient of electric potential,

  $$\boldsymbol{n} \cdot \nabla\phi(x, y, z)|_{Boundary} = -\sigma/\epsilon_r\epsilon_o.$$

  In the dimensionless expression,

  $$\boldsymbol{n} \cdot \nabla\phi(x, y, z)|_{Boundary} = -q/\mathsf{R}.$$

  where definition of dimensionless parameters are given by equations (1.42), and (1.43).

- **Constant potential on a boundary (Dirichlet boundary condition)**
  On a boundary surface (or generally in a partial region), the electric potential can be set to a constant value ϕ_o:

  $$\phi(x, y, z)|_{Boundary} = \phi_o \qquad (= \text{Constant}).$$

- **Zero-current condition on a boundary**(Neumann condition)
  Requiring an integrated electric current on a boundary should be zero, and the gradient of the electric potential got by this condition is applied as the Neumann condition. See section 1.6.3 in detail.

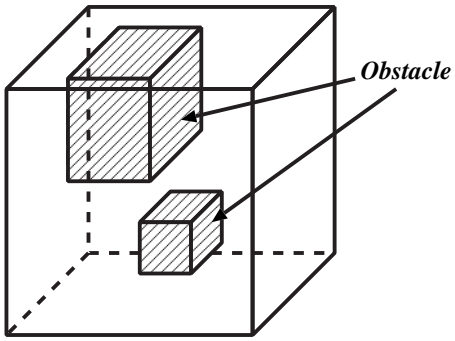

Figure 1.1: A system which has two obstacles.

Chapter 2

Sample problems of Electrolyte

2.1 Sample problems of Electrolyte_FDM

This section shows the applications for electrolyte simulator - Electrolyte_FDM - using the finite difference method. Input and output UDF files corresponding to these applications are in the sub-directory of `MUFFIN3/sample/ELECTROLYTE_FDM` according to the problems, namely in the directories `MUFFIN3/sample/ELECTROLYTE_FDM/EX01, EX02, ...` etc.

2.1.1 Application 1: Electrolyte between electrodes

This calculation simulates an electric field and ion concentration between two electrodes on two Z-direction boundaries as a one-dimensional problem.

1. Mesh parameters: Set the parameter.mesh_parameter.type to "SIMPLERECTANGULAR". In order to create a one-dimensional mesh structure, input the parameter.mesh_parameter.axes[] as follows.

axes[]	values[]	input
[0]	[0]	0.0
[0]	[1]	0.0
[0]	[2]	0.0
[1]	[0]	0.0
[1]	[1]	0.0
[1]	[2]	0.0
[2]	[0]	0.0
[2]	[1]	32.0
[2]	[2]	64.0

2. Solver parameter: Input parameters for an electric field solver.

Solver parameter	input
MAX_ITERATION_FOR_E-POTENTIAL_SOLVER	1.0e6
MONITORING_INTERVAL_OF_E-POTENTIAL_SOLVER	100
CONVERGENCE_CRITERION_FOR_E-POTENTIAL	1.0e-4
ACCELERATION_VALUE_FOR_E-POTENTIAL	1.8

3. Common physical parameter:
   Input DT=1.0e-2, FINAL_STEP=2001, and INTERVAL_OF_MONITORING=100.
   INTERVAL_OF_UDF_OUTPUT is set to 2000, but its value is scheduled as 1 from step 1, and 2000 from step 2.

4. Physical parameter: Input physical constants as follows.

Parameter	input
LEES_EDWARDS_BC	0
Number_of_Components	2
Average_Of_Concentration	0.1, 0.1
Diffusion_Coefficient	1.0, 1.0
Z	1, -1
R	3.89e-2
M	4.33e-1
D	0.517
Electric_Potential_at_XY_plane_ZM	-1.0
Electric_Potential_at_XY_plane_ZP	1.0
POSITION_Y_OF_CUTTING_ZX_Plane	1

5. region_condition[]:
   For each field, input boundary conditions as follows.
   XY_BOUNDARY_PLANE_ZM_AND_ZP for ElectricPotential applies electric voltage between electrodes.

name_of_region	name_of_target	name_of_condition
YZ_BOUNDARY_PLANE_XM_AND_XP	Concentration	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	Concentration	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	Concentration	ZM_WALL__ZP_WALL
YZ_BOUNDARY_PLANE_XM_AND_XP	ElectricPotential	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	ElectricPotential	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	ElectricPotential	ZM_DIRICHLET__ ZP_DIRICHLET
YZ_BOUNDARY_PLANE_XM_AND_XP	K_Field	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	K_Field	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	K_Field	ZM_WALL__ZP_WALL

6. Field[]: Register fields.

registered_field	type	name_of_region	num_of_component	io_flag
Concentration	Scalar	ALL_VERTEX	$(Number_of_Components)	1
ElectricPotential	Scalar	ALL_VERTEX	1	1
Pressure	Scalar	ALL_VERTEX	1	1
Velocity	Vector	ALL_EDGE	3	0
K_Field	Vector	ALL_EDGE	$(Number_of_Components) $(*)$(3)	0

7. procedures_table_for_initialization[]: Apply a command “UNIFORM” for the Concentration field.

8. procedures_table_for_evolution[]:
   Apply commands for fields as,
   “ELECTRIC_POTENTIAL_SOLVER” for the field ElectricPotential,
   “SOLVE_DIFFUSION_EQUATION_WITHOUT_FLOW” for the field Concentration,
   and “GRADIENT_ELECTRO_CHEMICAL_POTENTIAL” for the field K_Field.

2.1.2 Application 2: Diffusion layer near charged plane

This calculation simulates an electric field and ion concentration near a charged plane on -Z boundary. +Z boundary is set as connecting to bulk region in which charge neutrality is established (i.e. plus and minus ion have the same fixed concentrations). The electric field near the charged plane decays exponentially with the Debye length.

1. Mesh parameters:
   Set the parameter.mesh_parameter.type to "SIMPLERECTANGULAR". In order to create a one-dimensional mesh structure with 64 meshes dividing 32 unit-lengths of electrode distance, input the parameter.mesh_parameter.axes[] as follows.

axes[]	values[]	input
[0]	[0]	0.0
[0]	[1]	0.0
[0]	[2]	0.0
[1]	[0]	0.0
[1]	[1]	0.0
[1]	[2]	0.0
[2]	[0]	0.0
[2]	[1]	32.0
[2]	[2]	64.0

2. Solver parameter: Input parameters for an electric field solver.

Solver parameter	input
MAX_ITERATION_FOR_E-POTENTIAL_SOLVER	1.0e6
CONVERGENCE_CRITERION_FOR_E-POTENTIAL	1.0e-4
ACCELERATION_VALUE_FOR_E-POTENTIAL	1.8
MONITORING_INTERVAL_OF_E-POTENTIAL_SOLVER	100
LEES_EDWARDS_BC	0

3. Common physical parameter:
   Input DT=5.0e-2, FINAL_STEP=10001, and INTERVAL_OF_MONITORING=100.
   INTERVAL_OF_UDF_OUTPUT is set to 2000, but its value is scheduled as 1 from step 1, and 2000 from step 2.

4. Physical parameter: Input physical constants as follows.

Parameter	input
Number_of_Components	2
Average_Of_Concentration	0.1,0.1
Diffusion_Coefficient	1.0, 1.0
Z	1, -1
R	3.89e-2
M	4.33e-1
D	0.517
GRADIENT_OF_E-POTENTIAL_AT_XY_PLANE_ZM	-1.0
Electric_Potential_at_XY_plane_ZP	0
POSITION_Y_OF_CUTTING_ZX_Plane	1

5. region_condition[]:
   For each field, input boundary conditions as follows.
   The setting of "*_PLANE_ZM_AND_ZP" for each field is essential in this case.

name_of_region	name_of_target	name_of_condition
YZ_BOUNDARY_PLANE_XM_AND_XP	Concentration	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	Concentration	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	Concentration	ZM_WALL__ZP_BULK
YZ_BOUNDARY_PLANE_XM_AND_XP	ElectricPotential	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	ElectricPotential	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	ElectricPotential	ZM_NEUMANN__ ZP_DIRICHLET
YZ_BOUNDARY_PLANE_XM_AND_XP	K_Field	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	K_Field	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	K_Field	ZM_WALL__ZP_BULK

6. Field[]: Register fields as follows.

registered_field	type	name_of_region	num_of_component	io_flag
Concentration	Scalar	ALL_VERTEX	$(Number_of_Components)	1
ElectricPotential	Scalar	ALL_VERTEX	1	1
Pressure	Scalar	ALL_VERTEX	1	1
Velocity	Vector	ALL_EDGE	3	1
K_Field	Vector	ALL_EDGE	$(Number_of_Components) $(*)$(3)	0

7. procedures_table_for_initialization[]:
   Apply a command "UNIFORM" for the Concentration field.

8. procedures_table_for_evolution[]:
   Apply commands for fields as,
   "ELECTRIC_POTENTIAL_SOLVER" for the field ElectricPotential,
   "SOLVE_DIFFUSION_EQUATION_WITHOUT_FLOW" for the field Concentration,
   and "GRADIENT_ELECTRO_CHEMICAL_POTENTIAL" for the field K_Field.

2.1.3 Application 3: Electrolyte flow with charged obstacle(1)

This calculation simulates a flow of an electrolyte solution around a charged "obstacle". The flow of electrolyte between two grounded electrodes (potential zero) on Z-direction boundaries is driven by a pressure gradient in the +X direction.

1. Mesh parameters:
   Set the parameter.mesh_parameter.type to "SIMPLERECTANGULAR". In order to make three-dimensional 32x32x32 meshes, input the parameter.mesh_parameter.axes[] as follows.

axes[]	values[]	input
[0]	[0]	0.0
[0]	[1]	31.0
[0]	[2]	31.0
[1]	[0]	0.0
[1]	[1]	31.0
[1]	[2]	31.0
[2]	[0]	0.0
[2]	[1]	31.0
[2]	[2]	31.0

2. Solver parameter: Input parameters for the electric field and velocity field solver.

Solver parameter	input
MAX_ITERATION_FOR_E-POTENTIAL_SOLVER	1.0e6
CONVERGENCE_CRITERION_FOR_E-POTENTIAL	1.0e-4
ACCELERATION_VALUE_FOR_E-POTENTIAL	1.5
MONITORING_INTERVAL_OF_E-POTENTIAL_SOLVER	100
Max_Iteration_For_Pressure_Solver	1.0e6
Convergence_Criterion_For_Pressure_Solver	1.0e-4
ACCELERATION_VALUE_FOR_PRESSURE_SOLVER	1.5
MONITORING_INTERVAL_OF_PRESSURE_SOLVER	100
Max_Iteration_For_Velocity_Solver	1.0e6
Convergence_Criterion_For_Velocity_Solver	1.0e-4
ACCELERATION_VALUE_FOR_VELOCITY_SOLVER	1.5
MONITORING_INTERVAL_OF_VELOCITY_SOLVER	100
SKIP_INTERVAL_VELOCITY_CALCULATION	1
LEES_EDWARDS_BC	0

3. Common physical parameter:
   Input DT=1.0e-2, FINAL_STEP=1000,
   INTERVAL_OF_MONITORING=1, and INTERVAL_OF_UDF_OUTPUT=100.

4. Physical parameter:
   Input physical constants as follows. The value of the Obstacle field is input from a file named "Obstacle.input" which includes "spherical" obstacle shape as non-zero field values.

Parameter	input
Number_of_Components	2
Average_Of_Concentration	0.1, 0.1
Diffusion_Coefficient	1.0, 1.0
Z	1, -1
R	3.89e-2
M	4.33e-1
D	0.517
Seed_of_Random_Number	1966715
Deviation_From_Average_Concentration	0.01
Electric_Potential_at_XY_plane_ZM	0
Electric_Potential_at_XY_plane_ZP	0
Pressure_Gradient	-0.01
SURFACE_CHARGE_ON_OBSTACLE	1.0
Position_Y_of_Cutting_ZX_plane	1
OBSTACLE_DATA_FILE	Obstacle.input

5. region_condition[]: For each field, input boundary conditions as follows.

name_of_region	name_of_target	name_of_condition
YZ_BOUNDARY_PLANE_XM_AND_XP	Concentration	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	Concentration	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	Concentration	ZM_WALL__ZP_WALL
YZ_BOUNDARY_PLANE_XM_AND_XP	ElectricPotential	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	ElectricPotential	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	ElectricPotential	ZM_DIRICHLET__ ZP_DIRICHLET
YZ_BOUNDARY_PLANE_XM_AND_XP	K_Field	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	K_Field	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	K_Field	ZM_WALL__ZP_WALL
YZ_BOUNDARY_PLANE_XM_AND_XP	Pressure	BIASED_PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	Pressure	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	Pressure	ZM_WALL__ZP_WALL
YZ_BOUNDARY_PLANE_XM_AND_XP	Velocity	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	Velocity	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	Velocity	ZM_WALL__ZP_WALL

6. Field[]: Register fields as follows.

registered_field	type	name_of_region	num_of_component	io_flag
Concentration	Scalar	ALL_VERTEX	$(Number_of_Components)	1
ElectricPotential	Scalar	ALL_VERTEX	1	1
Pressure	Scalar	ALL_VERTEX	1	1
Velocity	Vector	ALL_EDGE	3	1
K_Field	Vector	ALL_EDGE	$(Number_of_Components) $(*)$(3)	0
Obstacle	Scalar	ALL_VERTEX	1	1

7. procedures_table_for_initialization[]:
   Apply commands for fields as
   "READ_OBSTACLE_DATA" for the field Obstacle, and
   "UNIFORM_WITH_NOISE" and "SET_BOUNDARY_CONDITION" for the field Concentration.

8. procedures_table_for_evolution[]:
   Set a procedure table name to "ION_DISTRIBUTION_AROUND_A_CHARGED_OBSTACLE".
   Apply commands for fields as
   "ELECTRIC_POTENTIAL_SOLVER:OBSTACLE" for the field ElectricPotential,
   "GRADIENT_ELECTRO_CHEMICAL_POTENTIAL:OBSTACLE" for the field K_Field,
   "SOLVE_STOKES_EQUATION_AND_PRESSURE:OBSTACLE" for the field Velocity,
   and "SOLVE_DIFFUSION_EQUATION_WITHOUT_FLOW" for the field Concentration.

2.1.4 Application 4: Electrolyte flow with charged obstacle(2)

Simulate an electrolyte fluid flow through a hole in a charged "obstacle". The flow between two grounded electrodes (the potential is zero) on Z-direction boundaries is driven by a pressure gradient in the +X direction.

1. Mesh parameters:
   Set the parameter.mesh_parameter.type to "SIMPLERECTANGULAR". In order to make three-dimensional 64x8x32 meshes, input the parameter.mesh_parameter.axes[] as follows.

axes[]	values[]	input
[0]	[0]	0.0
[0]	[1]	63.0
[0]	[2]	63.0
[1]	[0]	0.0
[1]	[1]	7.0
[1]	[2]	7.0
[2]	[0]	0.0
[2]	[1]	31.0
[2]	[2]	31.0

2. Solver parameter: Input parameters for the electric field and velocity solver.

Solver parameter	input
MAX_ITERATION_FOR_E-POTENTIAL_SOLVER	1.0e6
CONVERGENCE_CRITERION_FOR_E-POTENTIAL	1.0e-4
ACCELERATION_VALUE_FOR_E-POTENTIAL	1.5
MONITORING_INTERVAL_OF_E-POTENTIAL_SOLVER	100
Max_Iteration_For_Pressure_Solver	1.0e6
Convergence_Criterion_For_Pressure_Solver	1.0e-4
ACCELERATION_VALUE_FOR_PRESSURE_SOLVER	1.5
MONITORING_INTERVAL_OF_PRESSURE_SOLVER	100
Max_Iteration_For_Velocity_Solver	1.0e6
Convergence_Criterion_For_Velocity_Solver	1.0e-4
ACCELERATION_VALUE_FOR_VELOCITY_SOLVER	1.5
MONITORING_INTERVAL_OF_VELOCITY_SOLVER	100
SKIP_INTERVAL_VELOCITY_CALCULATION	1

3. Common physical parameter:
Input DT=1.0e-2, FINAL_STEP=1000,
INTERVAL_OF_MONITORING=1, and INTERVAL_OF_UDF_OUTPUT=200.

4. Physical parameter: Input physical constants as follows. Value of Obstacle field is input from a file named “Obstacle.input” which includes obstacle shape as non-zero field values.

Parameter	input
LEES_EDWARDS_BC	0
N_Write_DATA	10
Number_of_Components	2
Average_Of_Concentration	0.1, 0.1
Diffusion_Coefficient	1.0, 1.0
Z	1, -1
R	3.89e-2
M	4.33e-1
D	0.517
Seed_of_Random_Number	1966715
Deviation_From_Average_Concentration	0.01
Electric_Potential_at_XY_plane_ZM	0
Electric_Potential_at_XY_plane_ZP	1
Pressure_Gradient	-0.01
SURFACE_CHARGE_ON_OBSTACLE	1.0
Position_Y_of_Cutting_ZX_plane	1
OBSTACLE_DATA_FILE	Obstacle.input

5. region_condition[]: For each field, input boundary conditions as follows.

name_of_region	name_of_target	name_of_condition
YZ_BOUNDARY_PLANE_XM_AND_XP	Concentration	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	Concentration	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	Concentration	ZM_WALL__ZP_WALL
YZ_BOUNDARY_PLANE_XM_AND_XP	ElectricPotential	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	ElectricPotential	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	ElectricPotential	ZM_DIRICHLET__ ZP_DIRICHLET
YZ_BOUNDARY_PLANE_XM_AND_XP	K_Field	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	K_Field	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	K_Field	ZM_WALL__ZP_WALL
YZ_BOUNDARY_PLANE_XM_AND_XP	Pressure	BIASED_PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	Pressure	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	Pressure	ZM_WALL__ZP_WALL
YZ_BOUNDARY_PLANE_XM_AND_XP	Velocity	PERIODIC
ZX_BOUNDARY_PLANE_YM_AND_YP	Velocity	PERIODIC
XY_BOUNDARY_PLANE_ZM_AND_ZP	Velocity	ZM_WALL__ZP_WALL

6. Field[]: Register fields as follows.

registered_field	type	name_of_region	num_of_component	io_flag
Concentration	Scalar	ALL_VERTEX	$(Number_of_Components)	1
ElectricPotential	Scalar	ALL_VERTEX	1	1
Pressure	Scalar	ALL_VERTEX	1	1
Velocity	Vector	ALL_EDGE	3	1
K_Field	Vector	ALL_EDGE	$(Number_of_Components) $(*)$(3)	0
Obstacle	Scalar	ALL_VERTEX	1	1

7. procedures_table_for_initialization[]:
   Apply commands for fields as
   "READ_OBSTACLE_DATA".
   for the field Obstacle, and
   "UNIFORM_WITH_NOISE and SET_BOUNDARY_CONDITION" for the field Concentration.

8. procedures_table_for_evolution[]:
   Apply commands for fields as
   "ELECTRIC_POTENTIAL_SOLVER:OBSTACLE" for the field ElectricPotential,
   "GRADIENT_ELECTRO_CHEMICAL_POTENTIAL:OBSTACLE" for the field K_Field,
   "SOLVE_STOKES_EQUATION_AND_PRESSURE:OBSTACLE" for the field Velocity, and
   "SOLVE_DIFFUSION_EQUATION_WITHOUT_FLOW". for the field Concentration.

2.2 Sample problems of Electrolyte_FEM

This section shows the applications for electrolyte simulator - Electrolyte_FEM - using the finite element method. Input and output UDF files corresponding to these applications are in sub-directory of `MUFFIN3/sample/ELECTROLYTE_FEM` according to the problems, namely in the directories `MUFFIN3/sample/ELECTROLYTE_FEM/EX01, EX02, ...` etc.

2.2.1 Application 1: Electrolyte between electrodes

Simulate an electric field and ion concentration between two electrodes on two Z-direction boundaries as a one-dimensional problem. The condition is similar to the application 1 of Electrolyte_FDM.

[Input UDF file]

`MUFFIN3/sample/ELECTROLYTE_FEM/EX01/EX01_in.udf`

[Explanation for input UDF]

- parameter.mesh_parameter:

  Mesh type is "UNSTRUCTURED_RECT", 2x2x64 meshes.

- parameter.physical_parameter[] :

  Parameters "R","M" and "D" are set as described in theory section. The "DIELECTRIC_CONSTANT" parameter is used to set the value of the DielectricConstant field (use value for the first component only). Setting a user defined parameter "C_infinity" to give an averaged concentration.

- region.condition[]

  Set the values ± 1 on the Z-direction boundaries as a boundary condition of the ElectricPotential.

- dynamics_manager.registered_field[]

  Register the Concentration, K_Field, ChargeDensity, DielectricConst and ElectricPotenatial fields. Do not register the Velocity and Pressure fields.

- dynamics_manager.procedures_table_for_initialization[]

  The DielectricConstant is set by a command "DIELECTRIC_CONSTANT_SETTING"
  which uses parameter "DIELECTRIC_CONSTANT" defined in parameter.physical_parameter[]. The Concentration field is set by a command "UNIFORM_CONCENTRATION" which uses a parameter "AVERAGED_ION_CONCENTRATION".

- dynamics_manager.procedures_table_for_evolution[]

  The ChargeDensityField is calculated by
  "CHARGE_DENSITY_DEPENDING_ON_ION_CONCENTRATION" using Concentration field and the parameter "Z", and the ElectricPotential field is calculated by using the ChargeDensity field.
  The K_Field is calculated by "GRADIENT_ELECTRO_CHEMICAL_POTENTIAL", and the Concentration field is calculated finally by "SOLVE_ELECTROLYTE_WITHOUT_FLOW".

2.2.2 Application 2: Diffusion layer near charged plane

Simulate an electric field and ion concentration near a charged plane on the -Z boundary. The +Z boundary is set as connecting to bulk region in which charge neutrality is established (i.e. plus and minus ions have the same constant concentrations). The electric field near the charged plane decays exponentially with the Debye length. The condition is similar to the application 2 of the Electrolyte_FDM.

[Input UDF file]

`MUFFIN3/sample/ELECTROLYTE_FEM/EX02/EX02_in.udf`

[Explanation for input UDF]

- parameter.mesh_parameter:

  Mesh type is "UNSTRUCTURED_RECT", 2x2x64 meshes.

- parameter.physical_parameter[] :

  Parameters "R","M" and "D" are set as described in theory section. The "DIELECTRIC_CONSTANT" parameter is used to set the value of the DielectricConstant field (use value for the first component only). Setting a user defined parameter "C_infinity" to set an averaged concentration. Another user defined parameter "e_dPhi_n" is used in setting a boundary condition of the Concentration and ElectricPotenatial fields.

- region.condition[]

  The Concentration field in Z-direction boundary "ZMAX" is set to a constant value for each ion component. The ElectricPotential field on "ZMIN" boundary is set by the Neumann condition using parameter "e_dPhi_n" to give gradient value -1, and set to zero on "ZMAX" boundary. The ElectricPotential on "ZMIN" boundary gives the ζ potential.

- dynamics_manager.registered_field[]

  Register the Concentration, K_Field,ChargeDensity, DielectricConst and ElectricPotential fields. Do not register the Velocity and Pressure fields.

- dynamics_manager.procedures_table_for_initialization[]

  The DielectricConstant is set by a command "DIELECTRIC_CONSTANT_SETTING"
  which uses a parameter "DIELECTRIC_CONSTANT" defined in the parameter.physical_parameter[].
  The Concentration field is set by a command "UNIFORM_CONCENTRATION" which uses a parameter "AVERAGED_ION_CONCENTRATION".

- dynamics_manager.procedures_table_for_evolution[]

  The ChargeDensityField is calculated by
  "CHARGE_DENSITY_DEPENDING_ON_ION_CONCENTRATION" using Concentration field and parameter "Z". The ElectricPotential field is calculated by using the ChargeDensity filed.
  The K_Field is calculated by "GRADIENT_ELECTRO_CHEMICAL_POTENTIAL". The Concentration field is calculated finally by "SOLVE_ELECTROLYTE_WITHOUT_FLOW".

2.2.3 Application 3: Electrophoresis for planar charged surface.

At first calculate an ion concentration and electric potential in Z-direction like the application 2 in an electrolyte, and applying a constant electric field in the X direction to induce a flow in the electrolyte. This is a simulation of an electrophoresis which will be approximated by the Smoluchowsky's equation[1]:

$$u_0 = \frac{\epsilon\epsilon_0\zeta}{\eta} E_0 \tag{2.1}$$

where u_0 is a electrophoresis velocity, $\epsilon\epsilon_0$ is the dielectric constant, ζ is the ζ-potential, η is the viscosity, and E_0 is the strength of the electric field.

[Input UDF file]

`MUFFIN3/sample/ELECTROLYTE_FEM/EX03/EX03_in.udf`

[Explanation for input UDF]

- parameter.mesh_parameter:

  Mesh type is "UNSTRUCTURED_RECT" with 3x2x64 meshes and periodic in the X-direction.

- parameter.solver_parameter:

  Set parameters "DT_FOR_V", "MAX_ITERATION_FOR_VELOCITY_SOLVER",
  and "CONVERGENCE_CRITERION_FOR_VELOCITY_SOLVER" for Stokes flow calculation.

- parameter.common_physical_parameter:

  Input DT=5.0e-2, FINAL_STEP=40000, and INTERVAL_OF_MONITORING=1.
  INTERVAL_OF_UDF_OUTPUT is set to 4000, but its value is scheduled as 1 from step 1, 500 from step 2, 10 from step 2000 and 4000 from step 2050.

- parameter.physical_parameter[] :

Parameter name(KEY)	value
NUMBER_OF_COMPONENTS	2
Z	1, -1
R	3.89e-2
M	4.33e-1
D	0.517
AVERAGED_ION_CONCENTRATION	0.1, 0.1
EXTERNAL_ELECTRIC_FIELD	1.0,0.0,0.0

- region.condition[]

  The Concentration field in Z-direction boundary "ZMAX" is set to be a constant value for each ion component.

  The ElectricPotential field in "ZMIN" boundary is set by the Neumann condition using parameter "e_dPhi_n" to give gradient value of -1, and set to zero on "ZMAX" boundary.

  The Velocity is fixed to zero on "ZMIN" boundary, and Y- and Z- components of Velocity on "ZMAX" are set to zero, but X-component is set free, so velocity X-component on "ZMAX" should give an electrophoresis velocity.

- dynamics_manager.registered_field[]

  Register all possible fields in this simulator (Concentration, K_Field, ChargeDensity, DielectricConst, ElectricPotenatial, Velocity and Pressure).

- dynamics_manager.procedures_table_for_initialization[]

  The DielectricConstant is set by a command "DIELECTRIC_CONSTANT_SETTING"
  and it uses a parameter "DIELECTRIC_CONSTANT" defined in the parameter.physical_parameter[].
  The Concentration field is set by a command "UNIFORM_CONCENTRATION" which uses a parameter "AVERAGED_ION_CONCENTRATION".

- dynamics_manager.procedures_table_for_evolution[].command_list[]

  Register two time evolution procedures "Solver A" and "Solver B". The procedure "Solver A" solves an ion diffusion process without flow. The procedure "Solver B" solves an electrolyte flow under a constant "external" electric field given by a parameter "EXTERNAL_ELECTRIC_FIELD". In the "Solver B", the K_Field is applied a command
  "GRADIENT_ELECTRO_CHEMICAL_POTENTIAL_WITH_EXTERNAL_FIELD"
  and the Velocity field is calculated as a Stokes flow. The calculation starts using "Solver A" and changes into "Solver B" from step 2000.

2.2.4 Application 4: Electrolyte flow around charged object(1)

A charged "particle" (Obstacle) is placed in an electrolyte flow induced by a pressure difference. By applying a "zero current" boundary condition, streaming potential will be induced in the direction of the flow.

[Input UDF file]

`MUFFIN3/sample/ELECTROLYTE_FEM/EX04/EX04_in.udf`

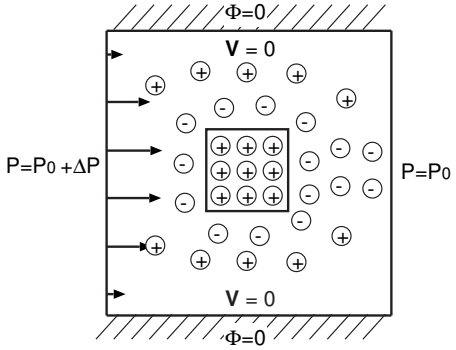

Figure 2.1: Electrolyte_FEM: Electrolyte flow

[Explanation for input UDF]

- parameter.mesh_parameter:

  Mesh type is "UNSTRUCTURED_RECT" with 32x32x2 meshes.

- parameter.solver_parameter:

  Set parameters "DT_FOR_V", "MAX_ITERATION_FOR_VELOCITY_SOLVER",
  and "CONVERGENCE_CRITERION_FOR_VELOCITY_SOLVER" for a Stokes flow calculation.

- parameter.common_physical_parameter[]:

  Input DT=0.1e-1, FINAL_STEP=600, INTERVAL_OF_MONITORING=1,
  and INTERVAL_OF_UDF_OUTPUT=100.

- parameter.physical_parameter[] :

Parameter name(KEY)	value
NUMBER_OF_COMPONENTS	2
Z	1, -1
R	3.89e-2
M	4.33e-1
D	0.517
DIELECTRIC_CONSTANT	(R)(/)$(M), 1
DIFFUSION_COEFFICIENT	1.0, 1.0
AVERAGED_ION_CONCENTRATION	0.1, 0.1
PARTICLE1_CHARGE	1.0
D_PRESSURE	3.0

  – The value of component zero of the parameter "DIELECTRIC_CONSTANT" is calculated from parameters "R" and "M". It is determined so that the dimensionless equation of electric field

$$\Delta\Phi = -\frac{\mathsf{M}}{\mathsf{R}} \sum_{\alpha} \mathsf{Z}_\alpha C_\alpha$$

  corresponds to the equation;

$$\Delta\Phi = -\frac{1}{\epsilon}\rho_e.$$

  The component zero of the parameter "DIELECTRIC_CONSTANT" is used to initialize the DielectricConst field by a command "CONSTANT_DIELECTRIC_CONSTANT", and the component one is not used in this case.

  – "D_PRESSURE" and "PARTICLE1_CHARGE" are user defined parameters, and used in partial region conditions region.condition[] for the Pressure and ChargeDensity, respectively.

- mesh.partial_region[]

  Two partial regions "particle_1" and "particle_1_cells" are given in input data. The region "particle_1" is given by node (vertices) IDs of a particle placed in electrolyte and the region "particle_1_cells" is given by IDs of elements (cells).

- region.condition[]

  Region conditions for the Velocity field on system boundaries are similar to those of the Poiseuille flow in the application 2 of the PhaseSeparation_FDM, so they are not explained here.

name	partial region	field	condition name	value
particle_1_vx	particle_1	Velocity	D_VX	0
particle_1_vy	particle_1	Velocity	D_VY	0
particle_1_vz	particle_1	Velocity	D_VZ	0
XMIN_P	BOUNDARY _VERTEX_XMIN	Pressure	D	3.0
XMAX_P	BOUNDARY _VERTEX_XMAX	Pressure	D	0
particle_1 _rho_e	particle_1	ChargeDensity	D	0.1
C_infinity_0	BOUNDARY _VERTEX_YMIN	Concentration	D_CONSTANT_VALUE _FOR_A_COMPONENT	0,0.1
C_infinity_1	BOUNDARY _VERTEX_YMIN	Concentration	D_CONSTANT_VALUE _FOR_A_COMPONENT	1,0.1
C_infinity_0	BOUNDARY _VERTEX_YMAX	Concentration	D_CONSTANT_VALUE _FOR_A_COMPONENT	0,0.1
C_infinity_1	BOUNDARY _VERTEX_YMAX	Concentration	D_CONSTANT_VALUE _FOR_A_COMPONENT	1,0.1
C_infinity_0	BOUNDARY _VERTEX_XMIN	Concentration	D_CONSTANT_VALUE _FOR_A_COMPONENT	0,0.1
C_infinity_1	BOUNDARY _VERTEX_XMIN	Concentration	D_CONSTANT_VALUE _FOR_A_COMPONENT	1,0.1
XMIN_phi	BOUNDARY _VERTEX_XMIN	ElectricPotential	D	0
XMAX_phi	BOUNDARY _FACE_XMAX	ElectricPotential	N_ZERO_ELECTRIC _CURRENT	0
Particle_attr	particle_1_cells	Obstacle	D_CONSTANT _VALUE_FOR _AN_INDEX	1

  - Conditions particle_1_vx, particle_1_vy and particle_1_vz set the velocity to zero on nodes on the partial region "particle_1". In this way, we can set a non-moving object (obstacle) in a flow.
  - The condition particle_1_rho_e sets electric charge on nodes of the partial region "particle_1".
  - The condition "Particle_attr" sets non zero value for the Obstacle field for cells on the partial region "particle_1_cells". This prevents time evolution process of the fields K_Field and Concentration on cells on which the value of the Obstacle field is not zero.
  - The Concentration field values on X- and Y- boundaries are all set to a constant (bulk) value except "XMAX". On the boundary "XMAX", a condition "N_ZERO_ELECTRIC_CURRENT" for the ElectricPotential is used to calculate an electric field by "streaming potential" and to apply the field (gradient of potential) as the Neumann boundary condition.

- dynamics_manager.registered_field[]

  Register all possible fields in this simulator (Concentration, K_Field, ChargeDensity, DielectricConst, ElectricPotenatial, Velocity and Pressure).

- dynamics_manager.procedures_table_for_initialization[].command_list[]

field	command
Pressure	SET_ZERO
Velocity	SET_ZERO
ChargeDensity	INITIALIZE_BY_PARTIAL_REGION_CONDITION
DielectricConst	CONSTANT_DIELECTRIC_CONSTANT
Concentration	UNIFORM_CONCENTRATION
Obstacle	SET_ZERO
Obstacle	APPLY_PARTIAL_REGION_CONDITION

  - The ChargeDensity field is initialized by a partial region condition "particle_1_rho_e" applied on a partial region "particle_1" where a constant charge is set on the region.
  - The Obstacle field is set to zero at first by a command "SET_ZERO", and set as an obstacle by a command "APPLY_PARTIAL_REGION_CONDITION" which applies the partial region condition "Particle_attr".

- dynamics_manager.procedures_table_for_evolution[].command_list[]

field	command
ChargeDensity	CHARGE_DENSITY_DEPENDING_ON_ION_CONCENTRATION
ChargeDensity	APPLY_PARTIAL_REGION_CONDITION
ElectricPotential	ELECTRIC_POTENTIAL_SOLVER
K_Field	GRADIENT_ELECTRO_CHEMICAL_POTENTIAL
Velocity	SOLVE_STOKES_EQUATION_AND_PRESSURE
Concentration	SOLVE_ELECTROLYTE_WITH_FLOW

  - The ChargeDensity field is, at first, calculated by the ion concentration, and by applying the partial region condition "particle_1_rho_e" as the next command, an electric charge of the particle is reset to a constant value.

2.2.5 Application 5: Electrolyte flow around charged object(2)

A charged sphere (surface of void region with a surface charge boundary condition) is placed in an electrolyte flow induced by a pressure difference.

[Input UDF file]

`MUFFIN3/sample/ELECTROLYTE_FEM/EX05/EX05_in.udf`

[Explanation for input UDF]

- parameter.mesh_parameter:

  The mesh type is "UNSTRUCTURED_INPUT". Actual mesh data is created by a mesh generator MILK3 from meshg_inp.udf which resides in the same directory as the input UDF file.

- parameter.solver_parameter:

  Setting parameters "DT_FOR_V", and "CONVERGENCE_CRITERION_FOR_VELOCITY_SOLVER" for a Stokes flow calculation.

- parameter.common_physical_parameter[]:

  Input DT=0.4e-2, FINAL_STEP=4000, INTERVAL_OF_MONITORING=1,
  and INTERVAL_OF_UDF_OUTPUT=500.

- parameter.physical_parameter[] :

Parameter name(KEY)	value
NUMBER_OF_COMPONENTS	2
Z	1, -1
R	3.89e-2
M	4.33e-1
D	0.517
DIFFUSION_COEFFICIENT	1.0, 1.0
AVERAGED_ION_CONCENTRATION	0.1, 0.1
PARTICLE1_CHARGE	1.0
PRESSURE_DIFFERENCE	5.0

- mesh_coordinate, mesh_element,partial_region[]:

  Mesh data are created by MILK3.

- region.condition[]

  Region conditions for the Velocity field on system boundaries are similar to those of the Poiseuille flow in the application 2 of the PhaseSeparation_FDM, so they are not explained here.

name	partial region	field	condition name	value
particle_1_vx	BOUNDARY_VERTEX _INTERNAL_SPHERE_0	Velocity	D_VX	0
particle_1_vy	BOUNDARY_VERTEX _INTERNAL_SPHERE_0	Velocity	D_VY	0
particle_1_vz	BOUNDARY_VERTEX _INTERNAL_SPHERE_0	Velocity	D_VZ	0
XMIN_P	BOUNDARY _VERTEX_XMIN	Pressure	D	5.0
XMAX_P	BOUNDARY _VERTEX_XMAX	Pressure	D	0
XMIN_phi	BOUNDARY _VERTEX_XMIN	ElectricPotential	D	0
XMAX_phi	BOUNDARY _VERTEX_XMAX	ElectricPotential	D	0
particle_1 _rho_s	BOUNDARY_FACE _INTERNAL_SPHERE_0	ElectricPotential	N	1.0

  - Conditions “particle_1_vx”, “particle_1_vy” and “particle_1_vz” set the velocity to zero on nodes on the surface of the void sphere.
  - The condition “particle_1_rho_s” sets electric charge on surface of the spherical void region.

Result of calculation

Figure 2.2.5 shows a calculated ChargeDensity field for the last UDF output step. This figure is drawn by the viewer in GOURMET using a python script MeshfieldShow.py described in MUFFIN tools section.

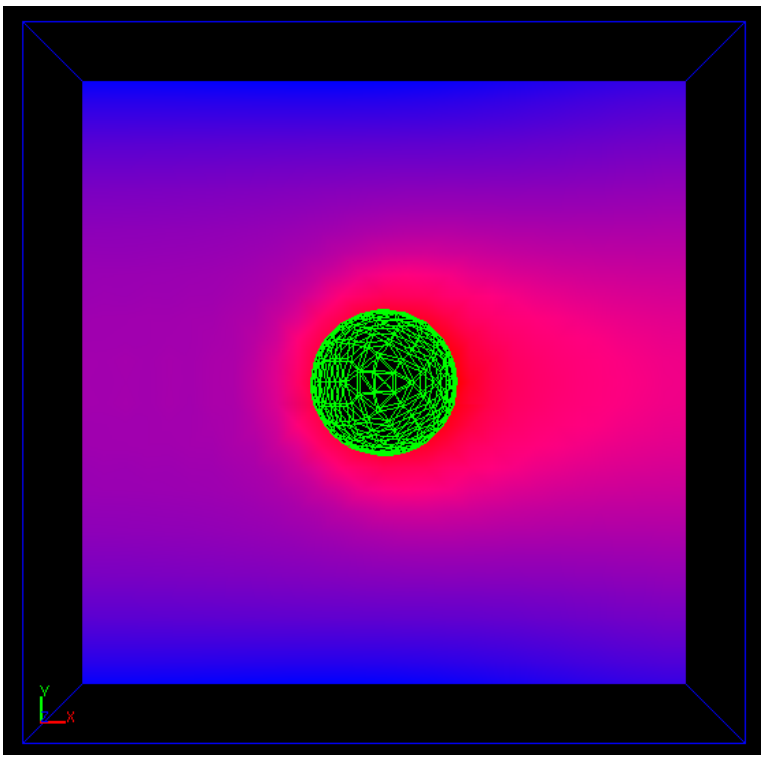

Figure 2.2: Electrolyte_FEM: A charge density distribution in an electrolyte flow around a charged spherical object.

Chapter 3

Operation guide of Electrolyte

3.1 Commands and parameters for fields of Electrolyte_FDM

3.1.1 Input parameters of Electrolyte_FDM

Name of Parameters	Meanings and notations in theory
NUMBER_OF_COMPONENTS	number of components
AVERAGED_CONCENTRATION	averaged concentration as initial value $C_{\alpha 0}$
DEVIATION_FROM _AVERAGED_CONCENTRATION	magnitude of noise given to initial value of concentration
SEED_OF_RANDOM_NUMBER	initial random number for random initialization of concentration field.
CONCENTRATION_GRADIENT _ALONG_X	X-direction gradient of concentration C_α for Biased Periodic Condition
CONCENTRATION_GRADIENT _ALONG_Y	Y-direction gradient of concentration C_α for Biased Periodic Condition
CONCENTRATION_GRADIENT _ALONG_Z	Z-direction gradient of concentration C_α for Biased Periodic Condition
BULK_CONCENTRATION	concentration of each component C_α for bulk boundary condition
POSITION_Y_OF_CUTTING_ZX_PLANE	Y coordinate index of XZ cutting plane for field value output in gnuplot format.
DIFFUSION_COEFFICIENT	diffusion coefficient of each component D_α
ACCELERATION_VALUE _FOR_E-POTENTIAL	acceleration factor for electric potential calculation
MAX_ITERATION _FOR_E-POTENTIAL_SOLVER	maximum number of iterations for electric potential calculation.
CONVERGENCE_CRITERION_FOR _E-POTENTIAL	convergence criterion of electric potential calculation
MONITORING_INTERVAL_OF _E-POTENTIAL_SOLVER	convergence monitoring interval of electric potential calculation
ELECTRIC_POTENTIAL_GRADIENT	initial value of an electric potential gradient
ELECTRIC_POTENTIAL _AT_YZ_PLANE_XM	boundary value of electric potential (YZ plane $-X$ side)
ELECTRIC_POTENTIAL _AT_YZ_PLANE_XP	boundary value of electric potential (YZ plane $+X$ side)
ELECTRIC_POTENTIAL _AT_ZX_PLANE_YM	boundary value of electric potential (ZX plane $-Y$ side)
ELECTRIC_POTENTIAL _AT_ZX_PLANE_YP	boundary value of electric potential (ZX plane $+Y$ side)
ELECTRIC_POTENTIAL	boundary value of electric potential

_AT_XY_PLANE_ZM	(XY plane $-Z$ side)
ELECTRIC_POTENTIAL _AT_XY_PLANE_ZP	boundary value of electric potential (XY plane $+Z$ side)
CONSTANT_TERM_OF _ELECTRIC_POTENTIAL _OSCILLATION_AT_YZ_PLANE_XP	ϕ_o of oscillating boundary electric potential; $\phi(x,y,z)\|_{Boundary} = \phi_o + \delta\phi \cdot \sin(\omega t)$ (YZ plane, $+X$ side)
AMPLITUDE_OF_ELECTRIC_POTENTIAL _OSCILLATION_AT_YZ_PLANE_XP	$\delta\phi$ of oscillating boundary electric potential (YZplane $+X$ side)
FREQUENCY_OF_ELECTRIC_POTENTIAL _OSCILLATION_AT_YZ_PLANE_XP	ω of oscillating boundary electric potential (YZplane, $+X$ side)
GRADIENT_OF_E-POTENTIAL _AT_YZ_PLANE_XM	boundary value of electric potential gradient (YZ plane, $-X$ side)
GRADIENT_OF_E-POTENTIAL _AT_YZ_PLANE_XP	boundary value of electric potential gradient (YZ plane, $+X$ side)
GRADIENT_OF_E-POTENTIAL _AT_ZX_PLANE_YM	boundary value of electric potential gradient (ZX plane, $-Y$ side)
GRADIENT_OF_E-POTENTIAL _AT_ZX_PLANE_YP	boundary value of electric potential gradient (ZX plane, $+Y$ side)
GRADIENT_OF_E-POTENTIAL _AT_XY_PLANE_ZM	boundary value of electric potential gradient (XY plane, $-Z$ side)
GRADIENT_OF_E-POTENTIAL _AT_XY_PLANE_ZP	boundary value of electric potential gradient (XY plane, $+Z$ side)
SURFACE_CHARGE_ON_OBSTACLE	surface charge of obstacle
Z	valence of each ion component Z_α
R	ratio of electrostatic and thermal energy $R = e\Phi_0/k_BT$
M	$\tilde{\mathsf{M}}$ (eq.(1.18))
D	$\tilde{\mathsf{D}}$ (eq.(1.20))
OBSTACLE_DATA_FILE	file name storing position of obstacle meshes
MONITORING_INTERVAL_OF _PRESSURE_SOLVER	convergence monitoring interval of pressure calculation
ACCELERATION_VALUE_FOR _PRESSURE_SOLVER	acceleration factor for pressure calculation
MAX_ITERATION_FOR _PRESSURE_SOLVER	maximum number of iterations for pressure calculation
CONVERGENCE_CRITERION _FOR_PRESSURE_SOLVER	convergence criterion of pressure calculation
PRESSURE_GRADIENT	boundary pressure gap for biased periodic condition of pressure
PRESSURE_AT_YZ_PLANE_XM	boundary pressure (YZ plane, $-X$ side)
PRESSURE_AT_YZ_PLANE_XP	boundary pressure (YZ plane, $+X$ side)
PRESSURE_AT_ZX_PLANE_YM	boundary pressure (ZX plane, $-Y$ side)
PRESSURE_AT_ZX_PLANE_YP	boundary pressure (ZX plane, $+Y$ side)
PRESSURE_AT_XY_PLANE_ZM	boundary pressure (XY plane, $-Z$ side)
PRESSURE_AT_XY_PLANE_ZP	boundary pressure (XY plane, $+Z$ side)
CONSTANT_VALUE_OF_P _OSCILLATION_AT _YZ_PLANE_XP	P_o of oscillating boundary pressure ; $P(x,y,z)\|_{Boundary} = P_o + \delta P \cdot \sin(\omega t)$ (YZplane, $+X$ side)
AMPLITUDE_OF_P _OSCILLATION_AT_YZ_PLANE_XP	δP of oscillating boundary pressure (YZplane, $+X$ side)
FREQUENCY_OF_P _OSCILLATION_AT_YZ_PLANE_XP	ω of oscillating boundary pressure (YZplane, $+X$ side)
VELOCITY_RAW_DATA_FILE	input file name for initialization of velocity by file input.
SKIP_INTERVAL	time interval for velocity calculation

_VELOCITY_CALCULATION	
VX_AT_YZ_PLANE_XM	X component of boundary velocity (YZ plane, $-X$ side)
VY_AT_YZ_PLANE_XM	Y component of boundary velocity (YZ plane, $-X$ side)
VZ_AT_YZ_PLANE_XM	Z component of boundary velocity (YZ plane, $-X$ side)
VX_AT_YZ_PLANE_XP	X component of boundary velocity (YZ plane, $+X$ side)
VY_AT_YZ_PLANE_XP	Y component of boundary velocity (YZ plane, $+X$ side)
VZ_AT_YZ_PLANE_XP	Z component of boundary velocity (YZ plane, $+X$ side)
VX_AT_ZX_PLANE_YM	X component of boundary velocity (ZX plane, $-Y$ side)
VY_AT_ZX_PLANE_YM	Y component of boundary velocity (ZX plane, $-Y$ side)
VZ_AT_ZX_PLANE_YM	Z component of boundary velocity (ZX plane, $-Y$ side)
VX_AT_ZX_PLANE_YP	X component of boundary velocity (ZX plane, $+Y$ side)
VY_AT_ZX_PLANE_YP	Y component of boundary velocity (ZX plane, $+Y$ side)
VZ_AT_ZX_PLANE_YP	Z component of boundary velocity (ZX plane, $+Y$ side)
VX_AT_XY_PLANE_ZM	X component of boundary velocity (XY plane, $-Z$ side)
VY_AT_XY_PLANE_ZM	Y component of boundary velocity (XY plane, $-Z$ side)
VZ_AT_XY_PLANE_ZM	Z component of boundary velocity (XY plane, $-Z$ side)
VX_AT_XY_PLANE_ZP	X component of boundary velocity (XY plane, $+Z$ side)
VY_AT_XY_PLANE_ZP	Y component of boundary velocity (XY plane, $+Z$ side)
VZ_AT_XY_PLANE_ZP	Z component of boundary velocity (XY plane, $+Z$ side)
MONITORING_INTERVAL_OF _VELOCITY_SOLVER	convergence monitoring interval of velocity calculation
ACCELERATION_VALUE_FOR _VELOCITY_SOLVER	acceleration factor for velocity calculation
CONVERGENCE_CRITERION_FOR _VELOCITY_SOLVER	convergence criterion of velocity calculation
MAX_ITERATION_FOR _VELOCITY_SOLVER	maximum number of iterations for velocity calculation

3.1.2 Electrolyte_FDM - list of fields

Field name	meanings and notation in theory
Concentration	Ion concentration field C_α
ElectricPotential	Electric potential fieldΦ
K_Field	Flux field $\boldsymbol{J}_\alpha$
Pressure	Pressure field P
Velocity	Velocity field $\boldsymbol{v}$
Obstacle	Obstacle field

3.1.3 Electrolyte_FDM - commands of fields

Concentration : Concentration field - commands

Concentration	Name
Initialization	"READ_AVS_DATA"
Initialization	"ADD_NOISE"
Initialization	"UNIFORM"
Initialization	"UNIFORM_WITH_NOISE"
Initialization	"LINEAR_ALONG_X_Direction"
Initialization	"LINEAR_ALONG_Y_Direction"
Initialization	"LINEAR_ALONG_Z_Direction"
Time evolution	"SOLVE_DIFFUSION_EQUATION_WITH_FLOW"
Time evolution	"SOLVE_DIFFUSION_EQUATION_WITHOUT_FLOW"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Analysis	"OUTPUT_SNAPSHOT_IN_GNP_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"
Evaluation	"TRUE_AT_A_CONSTANT_INTERVAL_TIME"

1. **Concentration - initialization commands**

Name	**"READ_AVS_DATA"**
Function	Read initial value from a file in AVS format.
Dependent parameter	AVS_DATA_FILE_NAME

Name	**"ADD_NOISE"**
Function	Add random noise.
Dependent filed	Obstacle
Dependent parameter	DEVIATION_FROM_AVERAGE_CONCENTRATION
Dependent parameter	SEED_OF_RANDOM_NUMBER

Name	**"UNIFORM"**
Function	Initialize concentration of each component to a constant value. Boundary values are set according to boundary conditions.
Dependent parameter	AVERAGE_OF_CONCENTRATION

Name	**"UNIFORM_WITH_NOISE"**
Function	Initialize concentration of each component to a constant value , and add noise. Boundary values are set according to boundary conditions.
Dependent filed	Obstacle
Dependent parameter	DEVIATION_FROM_AVERAGE_CONCENTRATION
Dependent parameter	SEED_OF_RANDOM_NUMBER
Dependent parameter	AVERAGE_OF_CONCENTRATION

Name	**"LINEAR_ALONG_X_Direction"**
Function	Initialize to have a constant gradient in X direction.
Dependent parameter	CONCENTRATION_GRADIENT_ALONG_X
Dependent parameter	AVERAGE_OF_CONCENTRATION

Name	**"LINEAR_ALONG_Y_Direction"**
Function	Initialize to have a constant gradient in Y direction.
Dependent parameter	CONCENTRATION_GRADIENT_ALONG_Y
Dependent parameter	AVERAGE_OF_CONCENTRATION

Name	**"LINEAR_ALONG_Z_Direction"**
Function	Initialize to have a constant gradient in Z direction.
Dependent parameter	CONCENTRATION_GRADIENT_ALONG_Y
Dependent parameter	AVERAGE_OF_CONCENTRATION

2. **Concentration - time evolution commands**

Name	**"SOLVE_DIFFUSION_EQUATION_WITH_FLOW"**
Function	One step time integration of equation for concentration field. $\frac{\partial C_\alpha}{\partial t} = -\nabla \cdot (\boldsymbol{v} C_\alpha) - \nabla \cdot \boldsymbol{J}_\alpha$
Dependent filed	Velocity
Dependent filed	K_Field
Dependent parameter	DT
Dependent parameter	CONCENTRATION_GRADIENT_ALONG_X
Dependent parameter	CONCENTRATION_GRADIENT_ALONG_Y
Dependent parameter	CONCENTRATION_GRADIENT_ALONG_Z
Dependent parameter	BULK_CONCENTRATION

Name	**"SOLVE_DIFFUSION_EQUATION_WITHOUT_FLOW"**
Function	One step time integration of equation for concentration field without flow field effect. $\frac{\partial C_\alpha}{\partial t} = -\nabla \cdot \boldsymbol{J}_\alpha$
Dependent filed	K_Field
Dependent parameter	DT
Dependent parameter	CONCENTRATION_GRADIENT_ALONG_X
Dependent parameter	CONCENTRATION_GRADIENT_ALONG_Y
Dependent parameter	CONCENTRATION_GRADIENT_ALONG_Z
Dependent parameter	BULK_CONCENTRATION

3. **Concentration - boundary condition (partial condition) commands**

Partial region condition	treatment
PERIODIC	periodic boundary condition
BIASED_PERIODIC	Biased Periodic boundary condition
XM_WALL__XP_WALL	wall boundary condition on both $-X$ and $+X$ boundaries.
XM_WALL__XP_BULK	wall boundary condition on $-X$, bulk boundary condition on $+X$.
XM_BULK__XP_WALL	bulk boundary condition on $-X$, wall boundary condition on $+X$.
XM_BULK__XP_BULK	bulk boundary condition on both $-X$ and $+X$ boundaries.
YM_WALL__YP_WALL	wall boundary condition on both $-Y$ and $+Y$ boundaries.
YM_WALL__YP_BULK	wall boundary condition on $-Y$, bulk boundary condition on $+Y$.
YM_BULK__YP_WALL	bulk boundary condition on $-Y$, wall boundary condition on $+Y$.
YM_BULK__YP_BULK	bulk boundary condition on both $-Y$ and $+Y$ boundaries.
ZM_WALL__ZP_WALL	wall boundary condition on both $-Z$ and $+Z$ boundaries.
ZM_WALL__ZP_BULK	wall boundary condition on $-Z$, bulk boundary condition on $+Z$.
ZM_BULK__ZP_WALL	bulk boundary condition on $-Z$, wall boundary condition on $+Z$.
ZM_BULK__ZP_BULK	bulk boundary condition on both $-Z$ and $+Z$ boundaries.

4. **Concentration - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(field-data).

Name	**"OUTPUT_SNAPSHOT_IN_GNP_FORMAT"**
Function	Output calculation results on an gnuplot format file.
Dependent parameter	POSITION_Y_OF_CUTTING_ZX_PLANE

5. **Concentration - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

Name	**"TRUE_AT_A_CONSTANT_TIME_INTERVAL"**
Function	Return "true" flag at a constant time interval.
Dependent parameter	FINAL_STEP
Dependent parameter	DIVISION_NUM1

ElectricPotential : electric potential field - commands

ElectricPotential	Name
Time evolution	"ELECTRIC_POTENTIAL_SOLVER"
Time evolution	"ELECTRIC_POTENTIAL_SOLVER:OBSTACLE"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT:E-FIELD"
Analysis	"OUTPUT_SNAPSHOT_IN_GNP_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"

1. **ElectricPotential - time evolution commands**

Name	**"ELECTRIC_POTENTIAL_SOLVER"**
Function	Calculate electric potential by solving Maxwell equation by iterative method (SOR).
Dependent parameter	ACCELERATION_VALUE_FOR_E-POTENTIAL
Dependent parameter	MAX_ITERATION_FOR_E-POTENTIAL_SOLVER
Dependent parameter	CONVERGENCE_CRITERION_FOR_E-POTENTIAL
Dependent parameter	MONITORING_INTERVAL_OF_E-POTENTIAL_SOLVER
Dependent parameter	ELECTRIC_POTENTIAL_GRADIENT
Dependent parameter	ELECTRIC_POTENTIAL_AT_YZ_PLANE_XM
Dependent parameter	ELECTRIC_POTENTIAL_AT_YZ_PLANE_XP
Dependent parameter	ELECTRIC_POTENTIAL_AT_ZX_PLANE_YM
Dependent parameter	ELECTRIC_POTENTIAL_AT_ZX_PLANE_YP
Dependent parameter	ELECTRIC_POTENTIAL_AT_XY_PLANE_ZM
Dependent parameter	ELECTRIC_POTENTIAL_AT_XY_PLANE_ZP
Dependent parameter	CONSTANT_TERM_OF_ELECTRIC_POTENTIAL_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	AMPLITUDE_OF_ELECTRIC_POTENTIAL_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	FREQUENCY_OF_ELECTRIC_POTENTIAL_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	GRADIENT_OF_E-Potential_AT_YZ_PLANE_XM
Dependent parameter	GRADIENT_OF_E-Potential_AT_YZ_PLANE_XP
Dependent parameter	GRADIENT_OF_E-Potential_AT_ZX_PLANE_YM
Dependent parameter	GRADIENT_OF_E-Potential_AT_ZX_PLANE_YP
Dependent parameter	GRADIENT_OF_E-Potential_AT_XY_PLANE_ZM
Dependent parameter	GRADIENT_OF_E-Potential_AT_XY_PLANE_ZP

Name	**"ELECTRIC_POTENTIAL_SOLVER:OBSTACLE"**
Function	Calculate electric potential by solving Maxwell equation by iterative method (SOR). Do not perform calculation on meshes on which Obstacle field has non-zero value, and apply surface charge boundary condition on obstacle surface.
Dependent filed	Obstacle
Dependent parameter	ACCELERATION_VALUE_FOR_E-POTENTIAL
Dependent parameter	MAX_ITERATION_FOR_E-POTENTIAL_SOLVER
Dependent parameter	CONVERGENCE_CRITERION_FOR_E-POTENTIAL
Dependent parameter	MONITORING_INTERVAL_OF_E-POTENTIAL_SOLVER
Dependent parameter	ELECTRIC_POTENTIAL_GRADIENT
Dependent parameter	ELECTRIC_POTENTIAL_AT_YZ_PLANE_XM
Dependent parameter	ELECTRIC_POTENTIAL_AT_YZ_PLANE_XP
Dependent parameter	ELECTRIC_POTENTIAL_AT_ZX_PLANE_YM
Dependent parameter	ELECTRIC_POTENTIAL_AT_ZX_PLANE_YP
Dependent parameter	ELECTRIC_POTENTIAL_AT_XY_PLANE_ZM
Dependent parameter	ELECTRIC_POTENTIAL_AT_XY_PLANE_ZP
Dependent parameter	CONSTANT_TERM_OF_ELECTRIC_POTENTIAL_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	AMPLITUDE_OF_ELECTRIC_POTENTIAL_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	FREQUENCY_OF_ELECTRIC_POTENTIAL_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	GRADIENT_OF_E-Potential_AT_YZ_PLANE_XM
Dependent parameter	GRADIENT_OF_E-Potential_AT_YZ_PLANE_XP
Dependent parameter	GRADIENT_OF_E-Potential_AT_ZX_PLANE_YM
Dependent parameter	GRADIENT_OF_E-Potential_AT_ZX_PLANE_YP
Dependent parameter	GRADIENT_OF_E-Potential_AT_XY_PLANE_ZM
Dependent parameter	GRADIENT_OF_E-Potential_AT_XY_PLANE_ZP
Dependent parameter	SURFACE_CHARGE_ON_OBSTACLE

2. **ElectricPotential - boundary condition (partial condition) commands**

Partial region condition	treatment
PERIODIC	periodic boundary condition
BIASED_PERIODIC	Biased Periodic boundary condition
XM_NEUMANN__XP_NEUMANN	Neumann condition on both $-X$ and $+X$ boundaries
XM_NEUMANN__XP_DIRICHLET	Neumann condition on $-X$, and Dirichlet condition on $+X$
XM_DIRICHLET__XP_NEUMANN	Dirichlet condition on $-X$, Neumann condition on $+X$
XM_DIRICHLET__XP_DIRICHLET	Dirichlet condition on both $-X$ and $+X$ boundaries
OSCILLATORY	oscillation potential boundary condition $\phi(x,y,z)\|_{Boundary} = \phi_o + \delta\phi \cdot \sin(\omega t)$
YM_NEUMANN__YP_NEUMANN	Neumann condition on both $-Y$ and $+Y$ boundaries
YM_NEUMANN__YP_DIRICHLET	Neumann condition on $-Y$, and Dirichlet condition on $+Y$
YM_DIRICHLET__YP_NEUMANN	Dirichlet condition on $-Y$, Neumann condition on $+Y$
YM_DIRICHLET__YP_DIRICHLET	Dirichlet condition on both $-Y$ and $+Y$ boundaries
ZM_NEUMANN__ZP_NEUMANN	Neumann condition on both $-Z$ and $+Z$ boundaries
ZM_NEUMANN__ZP_DIRICHLET	Neumann condition on $-Z$, and Dirichlet condition on $+Z$
ZM_DIRICHLET__ZP_NEUMANN	Dirichlet condition on $-Z$, Neumann condition on $+Z$
ZM_DIRICHLET__ZP_DIRICHLET	Dirichlet condition on both $-Z$ and $+Z$ boundaries

3. **ElectricPotential - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(field-data).
Dependent filed	Obstacle

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT:E-FIELD"**
Function	Output electric field $\boldsymbol{E} = -\nabla\phi$ on an AVS format file(field-data).

Name	**"OUTPUT_SNAPSHOT_IN_GNP_FORMAT"**
Function	Output calculation results on a gnuplot format file.
Dependent parameter	POSITION_Y_OF_CUTTING_ZX_PLANE

4. **ElectricPotential - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

K_Field : flux field - commands

K_Field	Name
Initialization	"SET_ZERO"
Time evolution	"GRADIENT_ELECTRO_CHEMICAL_POTENTIAL"
Time evolution	"GRADIENT_ELECTRO_CHEMICAL_POTENTIAL:OBSTACLE"

1. **K_Field - initialization commands**

Name	**"SET_ZERO"**
Function	Set field values to zero.

2. **K_Field - time evolution commands**

Name	**"GRADIENT_ELECTRO_CHEMICAL_POTENTIAL"**
Function	Calculate flux from concentration and electric potential; $\nabla C_\alpha + \mathsf{R}\ \mathsf{Z}_\alpha C_\alpha \nabla\Phi$
Dependent filed Dependent filed	Concentration ElectricPotential
Dependent parameter	R

Name	**"GRADIENT_ELECTRO_ CHEMICAL_POTENTIAL:OBSTACLE"**
Function	Calculate flux from concentration and electric potential; Apply boundary condition $\boldsymbol{n} \cdot \boldsymbol{K}_\alpha = 0$ on surface of obstacles, and do not calculate K_Field in obstacles.
Dependent filed Dependent filed Dependent filed	Obstacle Concentration ElectricPotential
Dependent parameter	R

3. **K_Field - boundary condition (partial condition) commands**

Partial region condition	treatment
PERIODIC	periodic boundary condition
XM_WALL__XP_WALL	wall boundary condition on both $-X$ and $+X$ boundaries.
XM_WALL__XP_BULK	wall boundary condition on $-X$, bulk boundary condition on $+X$.
XM_BULK__XP_WALL	bulk boundary condition on $-X$, wall boundary condition on $+X$.
XM_BULK__XP_BULK	bulk boundary condition on both $-X$ and $+X$ boundaries.
YM_WALL__YP_WALL	wall boundary condition on both $-Y$ and $+Y$ boundaries.
YM_WALL__YP_BULK	wall boundary condition on $-Y$, bulk boundary condition on $+Y$.
YM_BULK__YP_WALL	bulk boundary condition on $-Y$, wall boundary condition on $+Y$.
YM_BULK__YP_BULK	bulk boundary condition on both $-Y$ and $+Y$ boundaries.
ZM_WALL__ZP_WALL	wall boundary condition on both $-Z$ and $+Z$ boundaries.
ZM_WALL__ZP_BULK	wall boundary condition on $-Z$, bulk boundary condition on $+Z$.
ZM_BULK__ZP_WALL	bulk boundary condition on $-Z$, wall boundary condition on $+Z$.
ZM_BULK__ZP_BULK	bulk boundary condition on both $-Z$ and $+Z$ boundaries.

Obstacle : obstacle field - commands

Obstacle	Name
Initialization	"READ_OBSTACLE_DATA"

1. **Obstacle - initialization commands**

Name	**"READ_OBSTACLE_DATA"**
Function	Initialize Obstacle field by reading a file.
Dependent parameter	OBSTACLE_DATA_FILE

Pressure : pressure field - commands

Pressure	Name
Initialization	"SET_ZERO"
Time evolution	"SOLVE_PRESSURE"
Time evolution	"CALCULATE_SOURCE_FIELD"
Time evolution	"ONE_ITERATION"
Time evolution	"SOLVE_PRESSURE_WITH_PARTICLES"
Analysis	"OUTPUT_SNAPSHOT_IN_GNP_FORMAT"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"

1. **Pressure - initialization commands**

Name	**"SET_ZERO"**
Function	Set field values to zero.

2. **Pressure - time evolution commands**

Name	**"SOLVE_PRESSURE"**
Function	Solve pressure field by iterative solution of Poisson equation; $\nabla^2 p = \nabla \cdot [\nabla(\eta\{\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^t\})] + \nabla \cdot \boldsymbol{K}$
Dependent filed	Velocity
Dependent filed	K_Field
Dependent parameter	ACCELERATION_VALUE_FOR_PRESSURE_SOLVER
Dependent parameter	MAX_ITERATION_FOR_PRESSURE_SOLVER
Dependent parameter	CONVERGENCE_CRITERION_PRESSURE_SOLVER
Dependent parameter	MONITORING_INTERVAL_OF_PRESSURE_SOLVER
Dependent parameter	VISCOSITY
Dependent parameter	PRESSURE_GRADIENT
Dependent parameter	PRESSURE_AT_YZ_PLANE_XM
Dependent parameter	PRESSURE_AT_YZ_PLANE_XP
Dependent parameter	PRESSURE_AT_ZX_PLANE_YM
Dependent parameter	PRESSURE_AT_ZX_PLANE_YP
Dependent parameter	PRESSURE_AT_XY_PLANE_ZM
Dependent parameter	PRESSURE_AT_XY_PLANE_ZP
Dependent parameter	CONSTANT_TERM_OF_PRESSURE_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	AMPLITUDE_OF_PRESSURE_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	FREQUENCY_OF_PRESSURE_OSCILLATION _AT_YZ_PLANE_XP

Name	**"CALCULATE_SOURCE_FIELD"**
Function	Calculate source term of Poisson equation of pressure field.
Dependent filed	Velocity
Dependent filed	K_Field
Dependent parameter	CA

Name	**"ONE_ITERATION"**
Function	One iteration of solution for Poisson equation of pressure field.
Dependent parameter	ACCELERATION_VALUE_FOR_PRESSURE_SOLVER
Dependent parameter	MAX_ITERATION_FOR_PRESSURE_SOLVER
Dependent parameter	CONVERGENCE_CRITERION_FOR_PRESSURE_SOLVER
Dependent parameter	MONITORING_INTERVAL_OF_PRESSURE_SOLVER

Name	**"SOLVE_PRESSURE_WITH_PARTICLES"**
Function	One iteration of solution for Poisson equation of pressure field. Calculate only for meshes on which obstacle field has non zero value.
Dependent filed	Obstacle
Dependent parameter	PRESSURE_GRADIENT
Dependent parameter	PRESSURE_AT_YZ_PLANE_XM
Dependent parameter	PRESSURE_AT_YZ_PLANE_XP
Dependent parameter	CONSTANT_TERM_OF_P_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	AMPLITUDE_OF_P_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	FREQUENCY_OF_P_OSCILLATION _AT_YZ_PLANE_XP
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	ACCELERATION_VALUE_FOR_PRESSURE_SOLVER
Dependent parameter	MAX_ITERATION_FOR_PRESSURE_SOLVER
Dependent parameter	CONVERGENCE_CRITERION_FOR_PRESSURE_SOLVER
Dependent parameter	MONITORING_INTERVAL_OF_PRESSURE_SOLVER

3. **Pressure - boundary condition (partial condition) commands**

Partial region condition	treatment
PERIODIC	periodic boundary condition
BIASED_PERIODIC	Biased Periodic boundary condition
XM_WALL__XP_WALL	set pressure to make ψ field satisfy wall boundary condition on both $-X$, and $+X$ boundaries. (called "wall boundary condition" in this table)
XM_WALL__XP_PRESSURE_SET	wall boundary condition on $-X$, constant pressure on $+X$
XM_WALL__XP_VELOCITY_SET	wall boundary condition on $-X$, constant velocity on $+X$
XM_PRESSURE_SET__XP_WALL	constant pressure on $-X$, wall boundary condition on $+X$
XM_PRESSURE_SET__XP_PRESSURE_SET	constant pressure on both $-X$ and $+X$ boundaries.
XM_PRESSURE_SET__XP_VELOCITY_SET	constant pressure on $-X$, constant velocity on $+X$
XM_VELOCITY_SET__XP_WALL	constant velocity on $-X$, wall boundary condition on $+X$
XM_VELOCITY_SET__XP_PRESSURE_SET	constant velocity on $-X$, constant pressure on $+X$
XM_VELOCITY_SET__XP_VELOCITY_SET	constant velocity on both $-X$ and $+X$ boundaries
OSCILLATORY_BIASED_PERIODIC	oscillating pressure condition on $+X$, $P = 0$ on $-X$
YM_WALL__YP_WALL	wall boundary condition on both $-Y$, and $+Y$ boundaries.
YM_WALL__YP_PRESSURE_SET	wall boundary condition on $-Y$, constant pressure on $+Y$
YM_WALL__YP_VELOCITY_SET	wall boundary condition on $-Y$, constant velocity on $+Y$
YM_PRESSURE_SET__YP_WALL	constant pressure on $-Y$, wall boundary condition on $+Y$
YM_PRESSURE_SET__YP_PRESSURE_SET	constant pressure on both $-Y$ and $+Y$ boundaries.
YM_PRESSURE_SET__YP_VELOCITY_SET	constant pressure on $-Y$, constant velocity on $+Y$
YM_VELOCITY_SET__YP_WALL	constant velocity on $-Y$, wall boundary condition on $+Y$
YM_VELOCITY_SET__YP_PRESSURE_SET	constant velocity on $-Y$, constant pressure on $+Y$
YM_VELOCITY_SET__YP_VELOCITY_SET	constant velocity on both $-Y$ and $+Y$ boundaries
ZM_WALL__ZP_WALL	wall boundary condition on both $-Z$, and $+Z$ boundaries.
ZM_WALL__ZP_PRESSURE_SET	wall boundary condition on $-Z$, constant pressure on $+Z$
ZM_WALL__ZP_VELOCITY_SET	wall boundary condition on $-Z$, constant velocity on $+Z$
ZM_PRESSURE_SET__ZP_WALL	constant pressure on $-Z$, wall boundary condition on $+Z$
ZM_PRESSURE_SET__ZP_PRESSURE_SET	constant pressure on both $-Z$ and $+Z$ boundaries.
ZM_PRESSURE_SET__ZP_VELOCITY_SET	constant pressure on $-Z$, constant velocity on $+Z$
ZM_VELOCITY_SET__ZP_WALL	constant velocity on $-Z$, wall boundary condition on $+Z$
ZM_VELOCITY_SET__ZP_PRESSURE_SET	constant velocity on $-Z$, constant pressure on $+Z$
ZM_VELOCITY_SET__ZP_VELOCITY_SET	constant velocity on both $-Z$ and $+Z$ boundaries

4. **Pressure - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_GNP_FORMAT"**
Function	Output calculation results on a gnuplot format file.
Dependent parameter	POSITION_Y_OF_CUTTING_ZX_PLANE

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(field-data).
Dependent parameter	NUMBER_OF_COMPONENTS

5. **Pressure - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

Velocity : velocity field - commands

Velocity	Name
Initialization	"SET_ZERO"
Initialization	"READ_VELOCITY_RAWDATA"
Time evolution	"SOLVE_STOKES_EQUATION"
Time evolution	"SOLVE_STOKES_EQUATION_AND_PRESSURE"
Time evolution	"SOLVE_STOKES_EQUATION_AND_PRESSURE:OBSTACLE"
Analysis	"OUTPUT_SNAPSHOT_IN_RAW_FORMAT"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Analysis	"OUTPUT_SNAPSHOT_IN_GNP_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"

1. **Velocity - initialization commands**

Name	**"SET_ZERO"**
Function	Set field values to zero.

Name	**"READ_VELOCITY_RAWDATA"**
Function	Input initial value of velocity from a file. ("OUTPUT_SNAPSHOT_IN_RAW_FORMAT")
Dependent parameter	VELOCITY_RAW_DATA_FILE

2. **Velocity - time evolution commands**

Name	**"SOLVE_STOKES_EQUATION_AND_PRESSURE"**
Function	Solve Stokes flow equation for velocity and pressure; $-\nabla p + \nabla(\eta\{\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^t\}) + \boldsymbol{K} = 0$
Dependent filed	Pressure
Dependent parameter	SKIP_INTERVAL_VELOCITY_CALCULATION
Dependent parameter	VX_AT_YZ_PLANE_XM
Dependent parameter	VY_AT_YZ_PLANE_XM
Dependent parameter	VZ_AT_YZ_PLANE_XM
Dependent parameter	VX_AT_YZ_PLANE_XP
Dependent parameter	VY_AT_YZ_PLANE_XP
Dependent parameter	VZ_AT_YZ_PLANE_XP
Dependent parameter	VX_AT_ZX_PLANE_YM
Dependent parameter	VY_AT_ZX_PLANE_YM
Dependent parameter	VZ_AT_ZX_PLANE_YM

Dependent parameter	VX_AT_ZX_PLANE_YP
Dependent parameter	VY_AT_ZX_PLANE_YP
Dependent parameter	VZ_AT_ZX_PLANE_YP
Dependent parameter	VX_AT_XY_PLANE_ZM
Dependent parameter	VY_AT_XY_PLANE_ZM
Dependent parameter	VZ_AT_XY_PLANE_ZM
Dependent parameter	VX_AT_XY_PLANE_ZP
Dependent parameter	VY_AT_XY_PLANE_ZP
Dependent parameter	VZ_AT_XY_PLANE_ZP
Dependent parameter	SHEAR_RATE_XZ
Dependent parameter	MONITORING_INTERVAL_OF_VELOCITY_SOLVER
Dependent parameter	ACCELERATION_VALUE_FOR_VELOCITY_SOLVER
Dependent parameter	CONVERGENCE_CRITERION_FOR_VELOCITY_SOLVER
Dependent parameter	MAX_ITERATION_FOR_VELOCITY_SOLVER

Name	**"SOLVE_STOKES_EQUATION"**
Function	One iteration of velocity solution cycle for Stokes flow.
Dependent filed	Pressure
Dependent filed	Obstacle
Dependent parameter	SKIP_INTERVAL_VELOCITY_CALCULATION
Dependent parameter	VX_AT_YZ_PLANE_XM
Dependent parameter	VY_AT_YZ_PLANE_XM
Dependent parameter	VZ_AT_YZ_PLANE_XM
Dependent parameter	VX_AT_YZ_PLANE_XP
Dependent parameter	VY_AT_YZ_PLANE_XP
Dependent parameter	VZ_AT_YZ_PLANE_XP
Dependent parameter	VX_AT_ZX_PLANE_YM
Dependent parameter	VY_AT_ZX_PLANE_YM
Dependent parameter	VZ_AT_ZX_PLANE_YM
Dependent parameter	VX_AT_ZX_PLANE_YP
Dependent parameter	VY_AT_ZX_PLANE_YP
Dependent parameter	VZ_AT_ZX_PLANE_YP
Dependent parameter	VX_AT_XY_PLANE_ZM
Dependent parameter	VY_AT_XY_PLANE_ZM
Dependent parameter	VZ_AT_XY_PLANE_ZM
Dependent parameter	VX_AT_XY_PLANE_ZP
Dependent parameter	VY_AT_XY_PLANE_ZP
Dependent parameter	VZ_AT_XY_PLANE_ZP
Dependent parameter	ACCELERATION_VALUE_FOR_VELOCITY_SOLVER
Dependent parameter	CONVERGENCE_CRITERION_FOR_VELOCITY_SOLVER
Dependent parameter	MAX_ITERATION_FOR_VELOCITY_SOLVER
Dependent parameter	MONITORING_INTERVAL_OF_VELOCITY_SOLVER

Name	**"SOLVE_STOKES_EQUATION_AND_ AND_PRESSURE:OBSTACLE"**
Function	Solve Stokes flow equation for velocity and pressure; Do not perform calculation on meshes on which Obstacle has non zero value, and apply boundary condition $\boldsymbol{V} = 0$ on the surface of obstacles.
Dependent filed	Pressure
Dependent filed	Obstacle
Dependent parameter	SKIP_INTERVAL_VELOCITY_CALCULATION
Dependent parameter	VX_AT_YZ_PLANE_XM
Dependent parameter	VY_AT_YZ_PLANE_XM
Dependent parameter	VZ_AT_YZ_PLANE_XM
Dependent parameter	VX_AT_YZ_PLANE_XP
Dependent parameter	VY_AT_YZ_PLANE_XP
Dependent parameter	VZ_AT_YZ_PLANE_XP
Dependent parameter	VX_AT_ZX_PLANE_YM
Dependent parameter	VY_AT_ZX_PLANE_YM
Dependent parameter	VZ_AT_ZX_PLANE_YM
Dependent parameter	VX_AT_ZX_PLANE_YP
Dependent parameter	VY_AT_ZX_PLANE_YP
Dependent parameter	VZ_AT_ZX_PLANE_YP
Dependent parameter	VX_AT_XY_PLANE_ZM
Dependent parameter	VY_AT_XY_PLANE_ZM
Dependent parameter	VZ_AT_XY_PLANE_ZM
Dependent parameter	VX_AT_XY_PLANE_ZP
Dependent parameter	VY_AT_XY_PLANE_ZP
Dependent parameter	VZ_AT_XY_PLANE_ZP
Dependent parameter	ACCELERATION_VALUE_FOR_VELOCITY_SOLVER
Dependent parameter	CONVERGENCE_CRITERION_FOR_VELOCITY_SOLVER
Dependent parameter	MAX_ITERATION_FOR_VELOCITY_SOLVER
Dependent parameter	MONITORING_INTERVAL_OF_VELOCITY_SOLVER

3. **Velocity - boundary condition (partial condition) commands**

Partial region condition	treatment
PERIODIC	periodic boundary condition
XM_WALL__XP_WALL	set pressure to make ψ field satisfy wall boundary condition on both $-X$, and $+X$ boundaries. (called "wall boundary condition" in this table)
XM_WALL__XP_PRESSURE_SET	wall boundary condition on $-X$, constant pressure on $+X$
XM_WALL__XP_VELOCITY_SET	wall boundary condition on $-X$, constant velocity on $+X$
XM_PRESSURE_SET__XP_WALL	constant pressure on $-X$, wall boundary condition on $+X$
XM_PRESSURE_SET__XP_PRESSURE_SET	constant pressure on both $-X$ and $+X$ boundaries.
XM_PRESSURE_SET__XP_VELOCITY_SET	constant pressure on $-X$, constant velocity on $+X$
XM_VELOCITY_SET__XP_WALL	constant velocity on $-X$, wall boundary condition on $+X$
XM_VELOCITY_SET__XP_PRESSURE_SET	constant velocity on $-X$, constant pressure on $+X$
XM_VELOCITY_SET__XP_VELOCITY_SET	constant velocity on both $-X$ and $+X$ boundaries
YM_WALL__YP_WALL	wall boundary condition on both $-Y$, and $+Y$ boundaries.
YM_WALL__YP_PRESSURE_SET	wall boundary condition on $-Y$, constant pressure on $+Y$
YM_WALL__YP_VELOCITY_SET	wall boundary condition on $-Y$, constant velocity on $+Y$
YM_PRESSURE_SET__YP_WALL	constant pressure on $-Y$, wall boundary condition on $+Y$
YM_PRESSURE_SET__YP_PRESSURE_SET	constant pressure on both $-Y$ and $+Y$ boundaries.
YM_PRESSURE_SET__YP_VELOCITY_SET	constant pressure on $-Y$, constant velocity on $+Y$
YM_VELOCITY_SET__YP_WALL	constant velocity on $-Y$, wall boundary condition on $+Y$
YM_VELOCITY_SET__YP_PRESSURE_SET	constant velocity on $-Y$, constant pressure on $+Y$
YM_VELOCITY_SET__YP_VELOCITY_SET	constant velocity on both $-Y$ and $+Y$ boundaries
ZM_WALL__ZP_WALL	wall boundary condition on both $-Z$, and $+Z$ boundaries.
ZM_WALL__ZP_PRESSURE_SET	wall boundary condition on $-Z$, constant pressure on $+Z$
ZM_WALL__ZP_VELOCITY_SET	wall boundary condition on $-Z$, constant velocity on $+Z$
ZM_PRESSURE_SET__ZP_WALL	constant pressure on $-Z$, wall boundary condition on $+Z$
ZM_PRESSURE_SET__ZP_PRESSURE_SET	constant pressure on both $-Z$ and $+Z$ boundaries.
ZM_PRESSURE_SET__ZP_VELOCITY_SET	constant pressure on $-Z$, constant velocity on $+Z$
ZM_VELOCITY_SET__ZP_WALL	constant velocity on $-Z$, wall boundary condition on $+Z$
ZM_VELOCITY_SET__ZP_PRESSURE_SET	constant velocity on $-Z$, constant pressure on $+Z$
ZM_VELOCITY_SET__ZP_VELOCITY_SET	constant velocity on both $-Z$ and $+Z$ boundaries

4. **Velocity - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_RAW_FORMAT"**
Function	Output calculation results on a file.

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(field-data).

Name	**"OUTPUT_SNAPSHOT_IN_GNP_FORMAT"**
Function	Output calculation results on a gnuplot format file.
Dependent parameter	POSITION_Y_OF_CUTTING_ZX_PLANE

5. **Velocity - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

3.2 Commands and parameters for fields of Electrolyte_FEM

3.2.1 Input parameters of Electrolyte_FEM

Name of Parameters	Meanings and notations in theory
NUMBER_OF_COMPONENTS	number of components
VALENCY	valence of each ion component Z_α
Z	valence of each ion component Z_α
R	ratio of electrostatic and thermal energy $R = e\Phi_0/k_BT$
M	$\tilde{\mathsf{M}}$ (eq.(1.18))
D	$\tilde{\mathsf{D}}$ (eq.(1.20))
DIELECTRIC_CONSTANT	relative dielectric constants ϵ_α. Give for each components.
MATRIX_SOLVER	matrix solver name to be used in pressure solution. Either "ICCG" or "CG" Default is "ICCG".
MATRIX_SOLVER _FOR_ELECTRIC_FIELD	matrix solver name to be used for electric field. Either "ICCG" or "CG" Default is "ICCG".
PENALTY_NUMBER	A penalty number to handle Dirichlet condition (a very large number). The default value is 10^{13}.
GRAVITY_X	X component of external force on fluid.
GRAVITY_Y	Y component of external force on fluid.
GRAVITY_Z	Z component of external force on fluid.
EXTERNAL_ELECTRIC_FIELD	uniform external electric field vector.
REYNOLDS	Reynolds number.
DT_FOR_V	time step interval for Stokes flow calculation.
MAX_ITERATION_FOR _VELOCITY_SOLVER	maximum number of iterations for Stokes flow calculation
CONVERGENCE_CRITERION_FOR _VELOCITY_SOLVER	convergence criterion for Stokes flow calculation > 0 : monitor relative change of velocity (default:1.0^{-3}) < 0 : monitor absolute change of velocity (default:1.0^{-3})
VISCOSITY	viscosity coefficient of each component η_α
SKIP_INTERVAL _VELOCITY_CALCULATION	time step interval for velocity calculation
AVERAGED_ION_CONCENTRATION	averaged ion concentration as initial value $C_{\alpha 0}$
DEVIATION_FROM_AVERAGED _ION_CONCENTRATION	magnitude of noise given to initial value of ion concentration
SEED_OF_RANDOM_NUMBER	initial random number for random initialization of ion concentration field.
NUMBER_OF_DROPLETS	number of droplets placed in initialization of ion concentration.
RADIUS_OF_DROPLET	radius of each droplet placed in initialization of ion concentration.
X_COORDINATE_OF_DROPLET	X coordinate of each droplet placed in initialization of ion concentration.
Y_COORDINATE_OF_DROPLET	Y coordinate of each droplet placed in initialization of ion concentration.
Z_COORDINATE_OF_DROPLET	Z coordinate of each droplet placed in initialization of ion concentration.
DIFFUSION_COEFFICIENT	diffusion coefficient of each ion component D_α
UNIFORM_CHARGE_DENSITY	charge desity value for uniform distribution.

3.2.2 Electrolyte_FEM - list of fields

Name of Parameters	Meanings and notations in theory
ChargeDensity	Electric charge density field ρ_e
Concentration	Ion concentration field C_α
DielectricConst	Dielectric constant filed ϵ
ElectricPotential	Electric potential field Φ
K_Field	Flux field $\boldsymbol{J}_\alpha(\boldsymbol{K})$
Obstacle	Obstacle filed (defined on FEM cells)
Pressure	Pressure field P
Velocity	Velocity field $\boldsymbol{v}$

The Obstacle filed is defined on FEM cells, and all other fields are defined on vertex.

3.2.3 Electrolyte_FEM - commands

ChargeDensity : charge density field - commands

ChargeDensity	Name
Initialization	"INITIALIZE_BY_PARTIAL_REGION_CONDITION"
Initialization	"UNIFORM_CHARGE_DENSITY"
Time evolution	"SET_ZERO"
Time evolution	"CHARGE_DENSITY_DEPENDING_ON_ION_CONCENTRATION"
Time evolution	"APPLY_PARTIAL_REGION_CONDITION"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"

1. **ChargeDensity - initialization commands**

Name	**"INITIALIZE_BY_PARTIAL_REGION_CONDITION"**
Function	Initialize field using initialization partial region conditions ("I_xxx").

Name	**"UNIFORM_CHARGE_DENSITY"**
Function	Initialize by a uniform constant charge density.
Dependent parameter	UNIFORM_CHARGE_DENSITY

2. **ChargeDensity - time evolution commands**

Name	**"SET_ZERO"**
Function	Set field values to zero.

Name	**"CHARGE_DENSITY_DEPENDING_ON_ ION_CONCENTRATION"**
Function	Calculate charge density from ion concentration, $\rho_e = \sum_\alpha eZ_\alpha C_\alpha$
Dependent field	VolumeFraction
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	VALENCY (or Z)

Name	**"APPLY_PARTIAL_REGION_CONDITION"**
Function	Apply partial region conditions.

3. **ChargeDensity - partial region condition (boundary condition) commands**

Partial region condition	treatment
I_CONSTANT_VALUE	Initialize to a constant value.
D or D_CONSTANT_VALUE	set to a constant value (Dirichlet condition)

4. **ChargeDensity - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(ucd-data).
Dependent field	ChargeDensity

5. **ChargeDensity - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

DielectricConst : dielectric constant field - commands

DielectricConst	Name
Time evolution	"CONSTANT_DIELECTRIC_CONSTANT"
Time evolution	"DIELECTRIC_CONSTANT_SETTING"
Time evolution	"APPLY_PARTIAL_REGION_CONDITION"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"

1. **DielectricConst - time evolution commands**

Name	**"CONSTANT_DIELECTRIC_CONSTANT"**
Function	Set uniform dielectric constant with ϵ_0 (component 0).
Dependent parameter	DIELECTRIC_CONSTANT

Name	**"DIELECTRIC_CONSTANT_SETTING"**
Function	Set an uniform dielectric constant $\epsilon = \mathsf{M}/\mathsf{R}$
Dependent field	VolumeFraction
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	M
Dependent parameter	R

Name	**"APPLY_PARTIAL_REGION_CONDITION"**
Function	Apply partial region conditions.

2. **DielectricConst - partial region condition (boundary condition) commands**

Partial region condition	treatment
D or D_CONSTANT_VALUE	Set to a constant value (Dirichlet condition)

3. **DielectricConst - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(ucd-data).

4. **DielectricConst - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

ElectricPotential : electric potential field - commands

ElectricPotential	Name
Time evolution	"ELECTRIC_POTENTIAL_SOLVER"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"

1. **ElectricPotential - time evolution commands**

Name	**"ELECTRIC_POTENTIAL_SOLVER"**
Function	Solve Poisson equation for electric potential $\nabla \cdot (\epsilon \nabla \Phi) = -\rho_e$
Dependent field	DielectricConst
Dependent field	ChargeDensity
Dependent parameter	CHARGE_DENSITY
Dependent parameter	DIELECTRIC_CONSTANT
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	PENALTY_NUMBER
Dependent parameter	MATRIX_SOLVER_FOR_ELECTRIC_FIELD
Dependent parameter	MATRIX_SOLVER

2. **ElectricPotential - partial region condition (boundary condition) commands**

Partial region condition	treatment
I_CONSTANT_VALUE	initialize to a constant value.
D	set to a constant value (Dirichlet condition)
N	$\boldsymbol{n} \cdot (\epsilon \nabla \Phi) = \bar{\boldsymbol{D}} \cdot \boldsymbol{n}$: give surface charge density (Neumann condition)
N_ZERO_ELECTRIC_CURRENT	calculate $\boldsymbol{n} \cdot \nabla \Phi$ from zero electric current condition on a surface.

3. **ElectricPotential - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(ucd-data).

4. **ElectricPotential - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

K_Field : flux field - commands

K_Field	Name
Initialization	"SET_ZERO"
Initialization	"SET_CONSTANT_FORCE"
Time evolution	"SET_ZERO"
Time evolution	"GRADIENT_ELECTRO_CHEMICAL_POTENTIAL"
Time evolution	"GRADIENT_ELECTRO_CHEMICAL_POTENTIAL _WITH_EXTERNAL_FIELD"
Time evolution	"APPLY_PARTIAL_REGION_CONDITION"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"

1. **K_Field - initialization commands**

Name	**"SET_ZERO"**
Function	Set field values to zero.
Dependent parameter	NUMBER_OF_COMPONENTS

Name	**"SET_CONSTANT_FORCE"**
Function	Apply a constant external force.
Dependent parameter	DIMENSION_OF_SPACE
Dependent parameter	GRAVITY_X
Dependent parameter	GRAVITY_Y
Dependent parameter	GRAVITY_Z

2. **K_Field - time evolution commands**

Name	**"SET_ZERO"**
Function	Set field values to zero.
Dependent parameter	NUMBER_OF_COMPONENTS

Name	**"GRADIENT_ELECTRO_CHEMICAL_POTENTIAL"**
Function	Calculate flux for each ion component; $\boldsymbol{J}_\alpha = \nabla C_\alpha + \mathsf{R}\ \mathsf{Z}_\alpha C_\alpha \nabla\Phi$ on surfaces of elements on which Obstacle field has non zero value, boundary condition $\boldsymbol{n} \cdot \boldsymbol{J}_\alpha = 0$ is applied automatically.
Dependent field	VolumeFraction
Dependent field	ElectricPotential
Dependent field	Obstacle
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	VALENCY or Z
Dependent parameter	R

Name	**"GRADIENT_ELECTRO_CHEMICAL_POTENTIAL _WITH_EXTERNAL_FIELD"**
Function	Calculate flux for each ion component; $\boldsymbol{J}_\alpha = \nabla C_\alpha + \mathsf{R}\ \mathsf{Z}_\alpha C_\alpha (\nabla\Phi - \boldsymbol{E}_0)$ on surfaces of elements on which Obstacle field has non zero value, boundary condition $\boldsymbol{n} \cdot \boldsymbol{J}_\alpha = 0$ is applied automatically.
Dependent field	VolumeFraction
Dependent field	ElectricPotential
Dependent field	Obstacle
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	EXTERNAL_ELECTRIC_FIELD
Dependent parameter	VALENCY or Z
Dependent parameter	R

Name	**"APPLY_PARTIAL_REGION_CONDITION"**
Function	Apply partial region conditions.
Dependent parameter	NUMBER_OF_COMPONENTS

3. **K_Field - partial region condition (boundary condition) commands**

Partial region condition	treatment
D_CONSTANT_VALUE_FOR_A_COMPONENT	set flux of a component to a constant value (Dirichlet condition) give component index α, $\boldsymbol{J}_{\alpha x}$, $\boldsymbol{J}_{\alpha y}$ and $\boldsymbol{J}_{\alpha z}$ as data part.

4. **K_Field - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(ucd-data).

5. **K_Field - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

Obstacle : obstacle field - commands

Obstacle	Name
Initialization	"SET_ZERO"
Time evolution	"APPLY_PARTIAL_REGION_CONDITION"
Evaluation	"RETURN_TRUE_FUNC"

1. **Obstacle - initialization commands**

Name	**"SET_ZERO"**
Function	Set field values to zero.

2. **Obstacle - time evolution commands**

Name	**"APPLY_PARTIAL_REGION_CONDITION"**
Function	Apply partial region conditions.

3. **Obstacle - partial region condition (boundary condition) commands**

Partial region condition	treatment
D or D_CONSTANT_VALUE	set to a constant value (Dirichlet condition)

4. **Obstacle - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

Pressure : pressure field - commands

Pressure	Name
Initialization	"SET_ZERO"
Time evolution	"SOLVE_PRESSURE"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"

1. **Pressure - initialization commands**

Name	**"SET_ZERO"**
Function	Set field values to zero.

2. **Pressure - time evolution commands**

Name	**"SOLVE_PRESSURE"**
Function	Solve Poisson equation for pressure, $\nabla^2 p = \frac{1}{\Delta t}\nabla \cdot \boldsymbol{v}^*$
Dependent field	Pressure
Dependent field	Velocity
Dependent parameter	DT_FOR_V
Dependent parameter	DIMENSION_OF_SPACE
Dependent parameter	PENALTY_NUMBER
Dependent parameter	MATRIX_SOLVER

3. **Pressure - partial region condition (boundary condition) commands**

Partial region condition	treatment
I_CONSTANT_VALUE	initialize to a constant value.
D	set to a constant value (Dirichlet condition)
N	$\boldsymbol{n} \cdot \nabla P = \bar{P}_n$ (Neumann condition)

4. **Pressure - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(ucd-data).

5. **Pressure - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

Velocity : velocity field - commands

Velocity	Name
Initialization	"SET_ZERO"
Initialization	"SET_DIRICHLET_CONDITION"
Time evolution	"SOLVE_VELOCITY_AND_PRESSURE"
Time evolution	"SOLVE_STOKES_EQUATION_AND_PRESSURE"
Time evolution	"SET_DIRICHLET_CONDITION"
Evaluation	"RETURN_TRUE_FUNC"

1. **Velocity - initialization commands**

Name	**"SET_ZERO"**
Function	Set field values to zero.

Name	**"SET_DIRICHLET_CONDITION"**
Function	Initialize using Dirichlet boundary conditions in partial region conditions.
Dependent field	Velocity

2. **Velocity - time evolution commands**

Name	**"SOLVE_VELOCITY_AND_PRESSURE"**
Function	One time step evolution of velocity by Navier Stokes equation, $Re\frac{\partial \boldsymbol{v}}{\partial t} = -\nabla p + \nabla(\eta\{\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^t\}) + \boldsymbol{K}$ Default value of Re is 1.0.
Dependent field	Pressure
Dependent field	K_Field
Dependent parameter	DT
Dependent parameter	REYNOLDS
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	M
Dependent parameter	D

Name	**"SOLVE_STOKES_EQUATION_AND_PRESSURE"**
Function	Solve Stokes flow equation for velocity and pressure; $\nabla p = \nabla(\eta\{\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^t\}) + \boldsymbol{K}$
Dependent field	Pressure
Dependent field	Velocity
Dependent field	K_Field
Dependent parameter	DT
Dependent parameter	DT_FOR_V
Dependent parameter	SKIP_INTERVAL_VELOCITY_CALCULATION
Dependent parameter	MAX_ITERATION_FOR_VELOCITY_SOLVER
Dependent parameter	CONVERGENCE_CRITERION_FOR_VELOCITY_SOLVER
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	M
Dependent parameter	D

3. **Velocity - partial region condition (boundary condition) commands**

Name	**"SET_DIRICHLET_CONDITION"**
Function	Apply Dirichlet boundary conditions in partial region conditions. When you are using command "SOLVE_VELOCITY_AND_PRESSURE" or "SOLVE_STOKES_EQUATION_AND_PRESSURE", partial region conditions are applied automatically, so you may not need to use this command explicitly.

4. **Velocity - partial region condition (boundary condition) commands**

Partial region condition	treatment
D_VX	set v_x to a constant value (Dirichlet condition)
D_VY	set v_y to a constant value (Dirichlet condition)
D_VZ	set v_z to a constant value (Dirichlet condition)

5. **Velocity - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(ucd-data).

6. **Velocity - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

Concentration : ion concentration field - commands

Concentration	Name
Initialization	"INITIALIZE_BY_PARTIAL_REGION_CONDITION"
Initialization	"ADD_NOISE"
Initialization	"UNIFORM_CONCENTRATION"
Initialization	"SET_DROPLETS"
Time evolution	"APPLY_PARTIAL_REGION_CONDITION"
Time evolution	"SOLVE_ELECTROLYTE_WITH_FLOW"
Time evolution	"SOLVE_ELECTROLYTE_WITHOUT_FLOW"
Analysis	"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"
Evaluation	"RETURN_TRUE_FUNC"

1. **Concentration - initialization commands**

Name	**"INITIALIZE_BY_PARTIAL_REGION_CONDITION"**
Function	Initialize field using initialization partial region conditions ("I_xxx").
Dependent parameter	NUMBER_OF_COMPONENTS

Name	**"ADD_NOISE"**
Function	Add random noise.
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	DEVIATION_FROM_AVERAGED_ION_CONCENTRATION
Dependent parameter	SEED_OF_RANDOM_NUMBER

Name	**"UNIFORM_CONCENTRATION"**
Function	Initialize concentration of each ion component to a constant value.
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	AVERAGED_ION_CONCENTRATION

Name	**"SET_DROPLETS"**
Function	Put droplets with specified positions and radii (2-component system only)
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	NUMBER_OF_DROPLETS
Dependent parameter	RADIUS_OF_DROPLET
Dependent parameter	X_COORDINATE_OF_DROPLET
Dependent parameter	Y_COORDINATE_OF_DROPLET
Dependent parameter	Z_COORDINATE_OF_DROPLET

2. **Concentration - time evolution commands**

Name	**"APPLY_PARTIAL_REGION_CONDITION"**
Function	Apply partial region conditions.
Dependent parameter	NUMBER_OF_COMPONENTS

Name	**"SOLVE_ELECTROLYTE_WITH_FLOW"**
Function	One step time integration of equation for ion concentration $\frac{\partial C_\alpha}{\partial t} = -\nabla \cdot (\boldsymbol{v} C_\alpha) - \nabla \cdot \boldsymbol{J}_\alpha$
Dependent field	K_Field
Dependent field	Velocity
Dependent field	Obstacle
Dependent parameter	DT
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	DIFFUSION_COEFFICIENT

Name	**"SOLVE_ELECTROLYTE_WITHOUT_FLOW"**
Function	One step time integration of equation for ion concentration $\frac{\partial C_\alpha}{\partial t} = -\nabla \cdot \boldsymbol{J}_\alpha$
Dependent field	K_Field
Dependent field	Obstacle
Dependent parameter	DT
Dependent parameter	NUMBER_OF_COMPONENTS
Dependent parameter	DIFFUSION_COEFFICIENT

3. **Concentration - partial region condition (boundary condition) commands**

Partial region condition	treatment
I_CONSTANT_VALUE_FOR_A_COMPONENT	initialize ψ_α to a constant value. Give component index α and ψ_α as data part.
D_CONSTANT_VALUE_FOR_A_COMPONENT	set ψ_α to a constant value (Dirichlet condition). Give component index α and ψ_α as data part.

4. **Concentration - analysis commands**

Name	**"OUTPUT_SNAPSHOT_IN_AVS_FORMAT"**
Function	Output calculation results on an AVS format file(ucd-data).

5. **Concentration - evaluation commands**

Name	**"RETURN_TRUE_FUNC"**
Function	Always return "true" flag. This function is used to perform an analysis command with a constant time step interval.

References

1) W.B.Russel, D. and W.R.Schowalter, : *Colloidal Dispersions*, Cambridge University Press (1989).