Class CIPCSL2 (need to be updated)

# Governing equation



in unit .

# Discretization

## Spatial discretization

The convection term is discretized with CIP-CSL2 scheme [1].

## Discretization in time

First source term is taken into account.



Next, convection term is computed, and the color function is updated.



# Structure

Cipcsl2.h

|——cipcsl2 : constructor

|

|——advance : source term and convection term

| |——convection : convection term

| | |——CIPCSL\* : convection of each direction

| | |——ib\_ext\_scalar : extrapolate color function to solid

| | '——ib\_bdcond : B.C. for IB

| |

| |——color\_minmax : min and max of color function

| |

| |——redist : control sharpening

| | |——distfunc : distance function

| | | |——set\_iflag : flagging

| | | '——insert\_bc\_dist : B.C. for distance function

| | |——gradphic : normal vector at cell center

| | |——set\_alp : coefficient for sharpening

| | |——sharpen : sharpening

| | '——update\_node : update values at node points etc.

| |——ib\_ext\_scalar : extrapolate scalar to solid

| '——ib\_bdcond : B.C. for IB

|

'——tension : surface tension

|——distfunc : distance function

| |——set\_iflag : flagging

| '——insert\_bc\_dist : B.C. for distance function

|

|——sharpen : sharpen with ε = 1.0 × dx, meaning smearing

|——smear : smear color function

|

|——curv : curvature

| |——gradphic : normal vector at cell center

| |——bnd\_sym\_kappa : B.C. of symmetry for kappa (adjacent cells)

| |——bnd\_wall\_kappa : B.C. of wall for kappa (adjacent cells)

| |——insert\_bc\_kappa : B.C. plane for kappa (on plane)

| |——curv\_interface : interpolate curvature at interface cells

| | '——gradphic : normal vector computed from distance function

| '——curv\_interface\_ext : extrapolate curvature from interface to others

| |——set\_wflag : flagging, wall adjacent cell = -1001

| '——insert\_bc\_kappa : boundary condition for kappa

|

'——bdcurv : curvature at wall boundary

|——gradphi : normal vector at node point

|——wall\_norm : normal vector on wall considering contact angle

|——ib\_norm : normal vector on IB considering contact angle

|——set\_wflag2 : flagging for wall (ordinary cells = -1001)

|——bdcurv\_interface : interpolate curvature at interface cells

|——bdcurv\_interface\_ext : extrapolate curvature from interface to other in wall

| (crude code)

'——insert\_bc\_kappa : B.C. plane for kappa, except kappa

# Prototype etc.

## Create object

CIPCSL2 conc(c, g, uvw, time, solver);

## Initialization

conc.init();

## Solve governing equations

conc.advance();

## Member functions available in main function

conc.set\_nredist(1);

conc.set\_itsharpen(10);

conc.set\_globalSharpen();

conc.front\_minmax();

conc.totalvol();

# Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Form | Description | Unit |
| clr | Scalar | protected color function |  |
| dist | Scalar | signed distance function | m |
| fn, atmp1 | Scalar | temporary array, normal size (ni, nj, nk) |  |
| delrho  vel | Scalar | temporary array, larger than normal (ni+1, nj+1, nk+1) |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Scalar temporary variables

advance(): sclr

CIPCSL\*: fn, vel

redist(): sclr (**cover multiple functions: sharpen, update\_node**)

set\_alp(): fn, atmp, stmp

sharpen(): fn, atmp, stmp

update\_node(): fn

tension(): sclr (**cover multiple functions: smooth, curv, bdcurv**)

distfunc(): fn, atmp (store initial value), iflag

smooth(): fn

curv\_interface():

# Member functions

## insert\_bc\_gradphic(const Scalar & val)

In this function, gradient of *val* in adjacent cells next wall, symmetric and immersed boundary is computed. Referring Fig. 1, we want to calculate  without using  .



Fig. 1 Sample configuration for wall at k-min.













With respect to parallelization,  may be outside of the array, if the wall is immersed boundary.

 or ???

References

[1] T. Nakamura, R. Tanaka, T. Yabe, K. Takizawa, Exactly conservative semi-Lagrangian scheme for multi-dimensional hyperbolic equations with directional splitting technique, J. Comput. Phys., 174 (2001) 171-207.