stokesLS

Generated by Doxygen 1.8.13

Contents

Chapter 1

stokesLS

A two dimesional tool for simulating the dynamics of a fluid/fluid interface in low Reynolds flow.

Quickstart

After downloading the source files, use

```
make shearDrop_exe
```

to build all source files and an example problem. The example problem can be executed by

```
./shearDrop_exe <Nx> <Ny>
```

where *Nx*, *Ny* are the number of grid points to use in the *x* and *y* directions, respectively. On the first run, the user will be prompted to create a parameter file specifying the initial position/radius of the drop. Answer `y` and input drop position/radius. Note that the default domain size is `[-1,1]x[-1,1]`.

Details

See Documents/ for mathematical and program details.

Authors/Acknowledgements

Colton Bryant

Level set library provided by David Chopp.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 fluidInterface Class Reference

`#include <fluidInterface.hpp>`

Public Member Functions

- fluidInterface (const int nx, const int ny, const double xMin, const double xMax, const double yMin, const double yMax, const double sigma)
- void setST (const double sigma)
- void setDeltaWidth (const double eps)
- double getXMin ()
- double getYMin ()
- double getXMax ()
- double getYMax ()
- double getDeltaWidth ()
- int nameToIndx (const string slice)
- void setSpeed (const stokesGrid ∗sGrid)
- void upwindAdvance (const double dt)
- void dumpSlice (const string slice, const string filename)
- void initialize (const InitialFunc &f)
- void computeSurfaceTension ()
- void computeSurfaceTension (stokesGrid ∗sGrid)
- double distToInt (levelset::Bicubic ∗fit, const double xi, const double yj, const double xMin, const double xMax, const double yMin, const double yMax, double &xC, double &yC, bool &clean)
- template<typename T >
  int sgn (T val)
- ∼fluidInterface (void)

4.1.1 Detailed Description

A class to interact with Prof. Chopp's level set library. Used to compute surface tension forces, compute the velocity extension, and update the interface location.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 fluidInterface()

```
fluidInterface::fluidInterface (
            const int nx,
            const int ny,
            const double xMin,
            const double xMax,
            const double yMin,
            const double yMax,
            const double sigma )
```

Creates the level set grid for our model of the fluid/fluid interface. A map is used to name each "slice" of the level set data.
Inputs: nx, ny - discrete grid size
xMin, xMax, yMin, yMax - domain bounds
sigma - surface tension

4.1.2.2 ~fluidInterface()

```
fluidInterface::~fluidInterface (
            void  )
```

Deallocates the level set data

4.1.3 Member Function Documentation

4.1.3.1 computeSurfaceTension() [1/2]

```
void fluidInterface::computeSurfaceTension ( )
```

Computes the surface tension body force using a discrete delta function of width 2∗eps. Note: This is computed on the nodes of the level set grid and should later be interpolated to the staggered grid for use with the Stokes solver.

4.1.3.2 computeSurfaceTension() [2/2]

```
void fluidInterface::computeSurfaceTension (
            stokesGrid * sGrid )
```

Computes surface tension and interpolates it onto the staggered grid for use by the stokes solver
Inputs: sGrid - a stokesGrid object

4.1.3.3 distToInt()

```
double fluidInterface::distToInt (
            levelset::Bicubic * fit,
            const double xi,
            const double yj,
            const double xMin,
            const double xMax,
            const double yMin,
            const double yMax,
            double & xC,
            double & yC,
            bool & clean )
```

Uses Newtons method to find the closest point on the interface to a given grid point and the distance to it
Inputs: Bicubic∗ fit - pointer to a bicubic fit of the level set data near the interface
xi, yj - Location of the grid point we are computing the distance to
xMin, xMax, yMin, yMax - Corners of the grid cell of interest
xC, yC - Location of the root found by the Newton's method (passed by reference for use later)
clean - set to true if the newton's method converges in maxIts iterations AND the root found is inside the current grid cell
Return: distance to the interface

4.1.3.4 dumpSlice()

```
void fluidInterface::dumpSlice (
            const string slice,
            const string filename )
```

Writes a named slice of the level set data to file.
input: string slice - name of a slice of the level set data, string filename - filename where data is to be written
result: level set data for the given slice is written to file. See LevelSet/src/um2io.cpp for details of the output file.

4.1.3.5 getDeltaWidth()

```
double fluidInterface::getDeltaWidth ( )
```

4.1.3.6 getXMax()

```
double fluidInterface::getXMax ( )
```

Get domain bounds. Used this for debugging sometimes
inputs: none
return: xMax

4.1.3.7 getXMin()

```
double fluidInterface::getXMin ( )
```

Get domain bounds. Used this for debugging sometimes
inputs: none
return: xMin

4.1.3.8 getYMax()

```
double fluidInterface::getYMax ( )
```

Get domain bounds. Used this for debugging sometimes
inputs: none
return: yMax

4.1.3.9 getYMin()

```
double fluidInterface::getYMin ( )
```

Get domain bounds. Used this for debugging sometimes
inputs: none
return: yMin

4.1.3.10 initialize()

```
void fluidInterface::initialize (
            const InitialFunc & f )
```

Sets initial level set function data using an InitialFunc from the level set code
Inputs: InitialFunc f - object specifying interface shape from the level set code. Other shapes found in Level↩ Set/src/initfuncs/
Result: the "phi" slice of level set data is set to a signed distance function with zero level set on the given shape.

4.1.3.11 nameToIndx()

```
int fluidInterface::nameToIndx (
            const string slice )
```

Returns the "slice index" corresponding to a valid named slice. Slices are named by strings in the constructor.
input: string slice - name of a slice of the level set data
return: k-index corresponding to that slice in the level set code. Note: An invalid string will throw an error.

4.1.3.12 setDeltaWidth()

```
void fluidInterface::setDeltaWidth (
            const double eps )
```

Set the width of the discrete delta function used in computing surface tension as a body force.
inputs: double eps - width to use
result: mEps set to eps.

4.1.3.13 setSpeed()

```
void fluidInterface::setSpeed (
            const stokesGrid * sGrid )
```

Velocity extension routine. Extrapolates normal velocity at the interface to the rest of the domain by following grad phi. Note: This function assumes x and y derivatives of phi have already been computed and stored in "phi_x" and "phi_y" slices. This happens automatically if surface tension has been computed earlier in the timestep.
Inputs: stokesGrid sGrid - a stokesGrid object whose velocity data will be used to compute the interface speed
Result: the "speed" slice of the level set data contains the extensional velocity F and "phi" can now be updated.

4.1.3.14 setST()

```
void fluidInterface::setST (
            const double sigma )
```

Set the surface tension parameter.
inputs: double sigma - surface tension parameter
result: mSigma set to sigma.

4.1.3.15 sgn()

```
template<typename T >
int fluidInterface::sgn (
            T val )  [inline]
```

4.1.3.16 upwindAdvance()

```
void fluidInterface::upwindAdvance (
            const double dt )
```

Use upwinding to advance the interface according to speed stored in the "speed" slice
Inputs: dt - the timestep to use
result: "phi" data updated according to the upwind scheme

The documentation for this class was generated from the following files:

- fluidInterface.hpp
- fluidInterface.cpp

4.2 stokesGrid Class Reference

```
#include <stokesGrid.hpp>
```

Public Member Functions

- stokesGrid (const int nx, const int ny, const double xMin, const double xMax, const double yMin, const double yMax)
- void setUppWallVelocity (const double val)
- void setLowWallVelocity (const double val)
- void setOmegaU (const double val)
- void setOmegaP (const double val)
- void setTol (const double val)
- double xC (const int i) const
- double yC (const int j) const
- double xF (const int i) const
- double yF (const int j) const
- double u (const int i, const int j) const

- double v (const int i, const int j) const
- double p (const int i, const int j) const
- int uI (const int i, const int j) const
- int vI (const int i, const int j) const
- int pI (const int i, const int j) const
- double fx (const int i, const int j)
- double fy (const int i, const int j)
- void setFX (const int i, const int j, const double val)
- void setFY (const int i, const int j, const double val)
- void solve ()
- void makeResidualsHuge ()
- void zeroOutResiduals ()
- bool isConverged ()
- double uB (const double x, const double y) const
- double vB (const double x, const double y) const
- double pB (const double x, const double y) const
- double bilin (const double a11, const double a12, const double a21, const double a22, const double ifrac, const double jfrac) const
- void dumpFlowData (string uOut, string vOut, string pOut)
- void dumpFX (string fxOut)
- void dumpFY (string fyOut)
- void turnOnVerbose ()
- ~stokesGrid (void)

4.2.1 Detailed Description

A generic class for solving Stokes equations in 2D with periodic boundary conditions in x and no slip/no penetration conditions in y. Wall velocities and general body force terms can be applied.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 stokesGrid()

```
stokesGrid::stokesGrid (
            const int nx,
            const int ny,
            const double xMin,
            const double xMax,
            const double yMin,
            const double yMax )
```

Constructor for the Stokes grid. Specifies domain/grid size, allocates storage, and specifies default values for parameters.
Inputs: nx,ny - discrete grid size
xMin, xMax, yMin, yMax - bounds of the domain

4.2.2.2 ~stokesGrid()

```
stokesGrid::~stokesGrid (
            void  )
```

Deallocates storage.

4.2.3 Member Function Documentation

4.2.3.1 bilin()

```
double stokesGrid::bilin (
            const double a11,
            const double a12,
            const double a21,
            const double a22,
            const double ifrac,
            const double jfrac ) const
```

Generic bilinear interpolation routine.
inputs: a11, a12, a21, a22 - data given on four corners of a unit square
ifrac, jfrac - values between 0 and 1 specifying the location to interpolate to.
e.g. ifrac = 0.5, jfrac = 0.75 will interpolate to the point (0.5, 0.75)
return: interpolated value at the given location

4.2.3.2 dumpFlowData()

```
void stokesGrid::dumpFlowData (
            string uOut,
            string vOut,
            string pOut )
```

Writes velocity and pressure data to binary files.
Creates 3 output files, one for x-velocity, one for y-velocity, one for pressure
Each file contains the following data:
integer xLen: the number of grid points in x for this array
integer yLen: the number of grid points in y for this array
double data: an xLen∗yLen length array of the data
double coords: an xLen+yLen length array containing the x then y coordinates of nodes where this data was stored in the staggered grid.

inputs: string uOut, vOut, pOut - filenames for the data to be written to. result: data written as described to these files in binary form

4.2.3.3 dumpFX()

```
void stokesGrid::dumpFX (
            string fxOut )
```

Writes the x-coordinate of the body force. See dumpFlowData for format information.

4.2.3.4 dumpFY()

```
void stokesGrid::dumpFY (
            string fyOut )
```

Writes the y-coordinate of the body force. See dumpFlowData for format information.

4.2.3.5 fx()

```
double stokesGrid::fx (
            const int i,
            const int j )
```

Returns the x-component of the body force here
inputs: int i, int j - index of cell
return: x body force here

4.2.3.6 fy()

```
double stokesGrid::fy (
            const int i,
            const int j )
```

Returns the y-component of the body force here
inputs: int i, int j - index of cell
return: y body force here

4.2.3.7 isConverged()

```
bool stokesGrid::isConverged ( )
```

Check convergence of the SOR solver.

4.2.3.8 makeResidualsHuge()

```
void stokesGrid::makeResidualsHuge ( )
```

Sets residuals to large values to force us into the while loop in solve.

4.2.3.9 p()

```
double stokesGrid::p (
            const int i,
            const int j ) const
```

pressure value.
inputs: int i, int j - index of cell
return: pressure in the cell

4.2.3.10 pB()

```
double stokesGrid::pB (
            const double x,
            const double y ) const
```

Bilinear interpolation for the pressure.
Converts a given point (x,y) to an x-velocity cell of the staggered grid. Then uses bilin to interpolate. Note: due to lack of pressure data at y=ymin, y=ymax, pressure cannot be interpolated to the upper/lower walls.
inputs: double x,y - location in the domain
return: interpolated value of p(x,y)

4.2.3.11 pI()

```
int stokesGrid::pI (
            const int i,
            const int j ) const
```

Converts i,j index to index in flowData array for pressure
inputs: int i, int j - index of cell
return: index of this value in flowData

4.2.3.12 setFX()

```
void stokesGrid::setFX (
            const int i,
            const int j,
            const double val )
```

Sets x body force in cell i,j
inputs: int i, int j - index of cell; double val - value to set
result: x-component of body force at index i,j set to val

4.2.3.13 setFY()

```
void stokesGrid::setFY (
            const int i,
            const int j,
            const double val )
```

Sets y body force in cell i,j
inputs: int i, int j - index of cell; double val - value to set
result: y-component of body force at index i,j set to val

4.2.3.14 setLowWallVelocity()

```
void stokesGrid::setLowWallVelocity (
            const double val )
```

Sets the wall velocity at y=yMin. Should be called during problem setup.
Inputs: double val - the velocity you wish to set
Result: uLow set to val

4.2.3.15 setOmegaP()

```
void stokesGrid::setOmegaP (
            const double val )
```

sets the relaxation parameter used in the pressure update. should be set during problem setup.
inputs: double val - the parameter to be used
result: omegaP set to val

4.2.3.16 setOmegaU()

```
void stokesGrid::setOmegaU (
            const double val )
```

Sets the relaxation parameter used in solving the momentum equation. Should be set during problem setup.
Inputs: double val - the parameter to be used
Result: omegaU set to val

4.2.3.17 setTol()

```
void stokesGrid::setTol (
            const double val )
```

sets the tolerance for the SOR solver. should be set during problem setup.
inputs: double val - the parameter to be used
result: tol set to val

4.2.3.18 setUppWallVelocity()

```
void stokesGrid::setUppWallVelocity (
            const double val )
```

Sets the wall velocity at y=yMax. Should be called during problem setup.
Inputs: double val - the velocity you wish to set
Result: uHigh set to val

4.2.3.19 solve()

```
void stokesGrid::solve ( )
```

Uses a modified SOR method to solve the stokes equations.
Further details of the solve can be found in the mathematical documentation.

4.2.3.20 turnOnVerbose()

```
void stokesGrid::turnOnVerbose ( )
```

Turns on residual information for the Stokes solver
inputs: none
result: verbose set to true; any call of the "solve" function will write residual information to the terminal throughout the update

4.2.3.21 u()

```
double stokesGrid::u (
            const int i,
            const int j ) const
```

x-velocity value.
inputs: int i, int j - index of cell
return: x-component of velocity field

4.2.3.22 uB()

```
double stokesGrid::uB (
            const double x,
            const double y ) const
```

Bilinear interpolation for the x-velocity.
Converts a given point (x,y) to an x-velocity cell of the staggered grid. Then uses bilin to interpolate. Note: this function accounts for the BCs.
inputs: double x,y - location in the domain
return: interpolated value of u(x,y)

4.2.3.23 uI()

```
int stokesGrid::uI (
            const int i,
            const int j ) const
```

Converts i,j index to index in flowData array for x-velocity
inputs: int i, int j - index of cell
return: index of this value in flowData

4.2.3.24 v()

```
double stokesGrid::v (
            const int i,
            const int j ) const
```

y-velocity value.
inputs: int i, int j - index of cell
return: y-component of velocity field

4.2.3.25 vB()

```
double stokesGrid::vB (
            const double x,
            const double y ) const
```

Bilinear interpolation for the y-velocity.
Converts a given point (x,y) to an x-velocity cell of the staggered grid. Then uses bilin to interpolate. Note: this function accounts for the BCs.
inputs: double x,y - location in the domain
return: interpolated value of v(x,y)

4.2.3.26 vI()

```
int stokesGrid::vI (
            const int i,
            const int j ) const
```

Converts i,j index to index in flowData array for y-velocity
inputs: int i, int j - index of cell
return: index of this value in flowData

4.2.3.27 xC()

```
double stokesGrid::xC (
            const int i ) const
```

x coordinate of cell center.
inputs: int i - index of cell
return: x-coordinate of cell center

4.2.3.28 xF()

```
double stokesGrid::xF (
            const int i ) const
```

x coordinate of cell face.
inputs: int i - index of cell
return: x-coordinate of cell face

4.2.3.29 yC()

```
double stokesGrid::yC (
            const int j ) const
```

y coordinate of cell center.
inputs: int j - index of cell
return: y-coordinate of cell center

4.2.3.30 yF()

```
double stokesGrid::yF (
            const int j ) const
```

y coordinate of cell face.
inputs: int j - index of cell
return: y-coordinate of cell face

4.2.3.31 zeroOutResiduals()

```
void stokesGrid::zeroOutResiduals ( )
```

Sets all maximum residuals back to zero. Done before each SOR update.

The documentation for this class was generated from the following files:

- stokesGrid.hpp
- stokesGrid.cpp

Chapter 5

File Documentation

5.1 fluidInterface.cpp File Reference

```
#include "fluidInterface.hpp"
```
Include dependency graph for fluidInterface.cpp:

5.2 fluidInterface.hpp File Reference

```
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <cmath>
#include <string>
#include <map>
#include "LevelSet/level/uniformmesh2d.h"
#include "LevelSet/level/um2boundary.h"
#include "LevelSet/level/um2linear.h"
#include "LevelSet/level/um2xperiodic.h"
#include "LevelSet/level/initialfunc.h"
#include "stokesGrid.hpp"
```
Include dependency graph for fluidInterface.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class fluidInterface

5.3 README.md File Reference

5.4 shearDrop.cpp File Reference

```
#include <stdio.h>
#include <iostream>
```

```
#include <fstream>
#include <math.h>
#include <string>
#include "stokesGrid.hpp"
#include "LevelSet/level/initialfunc.h"
#include "LevelSet/level/initfuncs/circle.h"
#include "LevelSet/level/initfuncs/linearfunc.h"
#include "fluidInterface.hpp"
```

Include dependency graph for shearDrop.cpp:

Functions

- int main (int argc, char ∗argv[ ])

5.4.1 Function Documentation

5.4.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

5.5 stokesGrid.cpp File Reference

```
#include "stokesGrid.hpp"
```

Include dependency graph for stokesGrid.cpp:

5.6 stokesGrid.hpp File Reference

```
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <cmath>
#include <string>
```

Include dependency graph for stokesGrid.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class stokesGrid