- burstingBubble.c

Login Logout

Login / Get an account Logout

- view
- history

burstingBubble.c

Experiment:

We investigate the classical problem of bubble bursting at a liquid-gas interface, but now in the presence of a viscoplastic liquid medium. Here are the schematics of the problem. This simualtion will start from Figure 1(c).

On this page, I am presenting the code that we used to simulate the process shown in the above video. The results presented here are currently under review in Science Advances. For codes for all the cases included in the manuscript, please visit the GitHub repository. I did not upload all of them here to avoid repeatability.

Numerical code

Id 1 is for the sessile drop, and Id 2 is mobile/impacting drop.

```
#include "axi.h"
#include "navier-stokes/centered.h"
#define FILTERED
#include "two-phaseVP.h"
#include "tension.h"
#include "reduced.h"
#include "distance.h"
#include "adapt_wavelet_limited.h"
```

```
#define tsnap (0.001)

// Error tolerancs
#define fErr (1e-3)                                 // error tolerance in f1 VOF
#define KErr (1e-4)                                 // error tolerance in f2 VOF
#define VelErr (1e-2)                               // error tolerances in velocity
#define OmegaErr (1e-3)                             // error tolerances in vorticity

// Numbers!
#define SIGMA 1.0
#define RHO21 0.001
#define MU21 0.02
#define Ldomain 8

// boundary conditions
u.n[right] = neumann(0.);
p[right] = dirichlet(0.);

int MAXlevel;
double La, Oh, Bond, tmax;
char nameOut[80], dumpFile[80];

int  main(int argc, char const *argv[]) {
  L0 = Ldomain;
  origin (-L0/2., 0.);
  init_grid (1 << 6);
  // Values taken from the terminal
  MAXlevel = atoi(argv[1]);
  tauy = atof(argv[2]);
  Bond = atof(argv[3]);
  La = atof(argv[4]);
  tmax = atof(argv[5]);
  if (argc < 6){
    fprintf(ferr, "Lack of command line arguments. Check! Need %d more arguments\n",5-argc)
    return 1;
  }
  fprintf(ferr, "Level %d, La %2.1e, Tauy %4.3f, Bo %4.3f\n", MAXlevel, La, tauy, Bond);
  char comm[80];
  sprintf (comm, "mkdir -p intermediate");
  system(comm);
  sprintf (dumpFile, "dump");
  Oh = sqrt(1./La);
  mumax = 1e8;  // The regularisation value of viscosity
  rho1 = 1., rho2 = RHO21;
  mu1 = Oh, mu2 = MU21*Oh;
  f.sigma = SIGMA;
  G.x = -Bond;
  run();
}

int refRegion(double x, double y, double z){
  return (y < 1.28 ? MAXlevel+2 : y < 2.56 ? MAXlevel+1 : y < 5.12 ? MAXlevel : MAXlevel-1)
}

event init (t = 0) {
  if (!restore (file = dumpFile)){

    char filename[60];
    sprintf(filename,"Bo%5.4f.dat",Bond);
    FILE * fp = fopen(filename,"rb");
    if (fp == NULL){
      fprintf(ferr, "There is no file named %s\n", filename);
      return 1;
    }
    coord* InitialShape;
    InitialShape = input_xy(fp);
    fclose (fp);
    scalar d[];
    distance (d, InitialShape);
    while (adapt_wavelet_limited ((scalar *){f, d}, (double[]){1e-8, 1e-8}, refRegion).nf);
```

The distance function is defined at the center of each cell, we have to calculate the value of this function at each vertex.

```
    vertex scalar phi[];
    foreach_vertex(){
      phi[] = -(d[] + d[-1] + d[0,-1] + d[-1,-1])/4.;
    }
```

We can now initialize the volume fraction of the domain.

```
    fractions (phi, f);
  }
}

scalar omega[];
event adapt(i++)
{
  scalar KAPPA[];
  curvature(f, KAPPA);
  vorticity (u, omega);
  foreach(){
    omega[] *= f[];
  }
  boundary ((scalar *){KAPPA, omega});
  if (t < 10*tsnap){
     adapt_wavelet_limited ((scalar *){f, KAPPA},
     (double[]){fErr, KErr}, refRegion);
   } else {
     adapt_wavelet_limited ((scalar *){f, u.x, u.y, KAPPA, omega},
     (double[]){fErr, VelErr, VelErr, KErr, OmegaErr},
     refRegion);
   }
}

// Outputs
event writingFiles (t = 0; t += tsnap; t <= tmax) {
  dump (file = dumpFile);
  sprintf (nameOut, "intermediate/snapshot-%5.4f", t);
  dump(file=nameOut);
}

event end (t = end) {
  fprintf(ferr, "Done: Level %d, La %2.1e, Tauy %4.3f, Bo %4.3f\n", MAXlevel, La, tauy, Bonc
}

event logWriting (i+=100) {
  double ke = 0.;
  foreach (reduction(+:ke)){
    ke += (2*pi*y)*(0.5*(f[])*(sq(u.x[]) + sq(u.y[])))*sq(Delta);
  }
  static FILE * fp;
  if (i == 0) {
    fprintf (ferr, "i dt t ke\n");
    fp = fopen ("log", "w");
    fprintf (fp, "i dt t ke\n");
    fprintf (fp, "%d %g %g %g\n", i, dt, t, ke);
    fclose(fp);
  } else {
    fp = fopen ("log", "a");
    fprintf (fp, "%d %g %g %g\n", i, dt, t, ke);
    fclose(fp);
  }
  fprintf (ferr, "%d %g %g %g\n", i, dt, t, ke);
  if (ke > 1e3 || ke < 1e-6){
    if (i > 1e2){
      return 1;
    }
  }
```

```
}
```

inspired by gitit, powered by darcsit