Rotating Reference Frame

Benny Smith

August 2021

1 Introduction

The reactor is represented as an ellipse that rocks back and forth with some constant maximum angle of rotation, such as 7°(as in literature?). Sketch below:

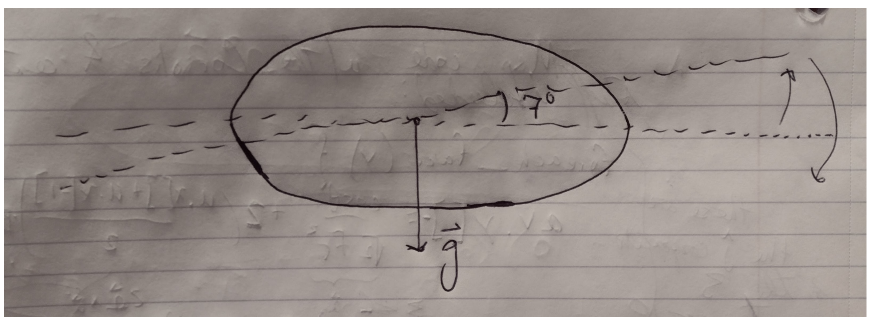

Figure 1: Rocking Ellipse

The centered Navier-Stokes discretization in Basilisk is in an inertial reference frame, so I must include extra terms having to do with coriolis and centripetal forces to get the accelerations correct.

2 Derivation of Acceleration Terms

The relationship between accelerations in the inertial and rotating reference frames is represented by the following equation [1]:

$$\boldsymbol{a}_F = \boldsymbol{a}_R + 2\boldsymbol{\Omega} \times \boldsymbol{u}_R - \Omega^2 \boldsymbol{R} \tag{1}$$

Where the F and R subscripts represent the inertial and rotational reference frames, respectively, $\boldsymbol{a}$ is acceleration, $\boldsymbol{u}$ is velocity, $\boldsymbol{\Omega}$ is angular velocity, and $\boldsymbol{R}$ is a position vector. The $2\boldsymbol{\Omega} \times \boldsymbol{u}_R$ term is a Coriolis acceleration, and the $-\Omega^2 \boldsymbol{R}$ is centripetal acceleration.

The acceleration event in Basilisk handles the x and y components separately, so I will break these vectors into their components here. But first I must consider the rocking motion. I want the rocking to be sinusoidal, so that Ω is maximized when the reactor's major axis is horizontal and minimized when the major axis is offset by some maximum angle Θ_{max} from the horizontal. And I want there to be some set period T. Thus the following function $\theta(t)$ is appropriate:

$$\theta(t) = \Theta_{max} sin(\frac{2\pi}{T}t) \tag{2}$$

The rotational speed Ω is the time derivative of $\theta(t)$:

$$\Omega(t) = \frac{2\pi\Theta_{max}}{T} cos(\frac{2\pi}{T}t) \tag{3}$$

So the vector $\mathbf{\Omega} = \frac{2\pi\Theta_{max}}{T} cos(\frac{2\pi}{T}t)\hat{\mathbf{k}}$ (can be positive or negative depending on the cosine).

The vector quantity $\boldsymbol{a}_R$ represents acceleration due to gravity in the rotating reference frame. This vector must have vertical and horizontal components due to the rocking motion:

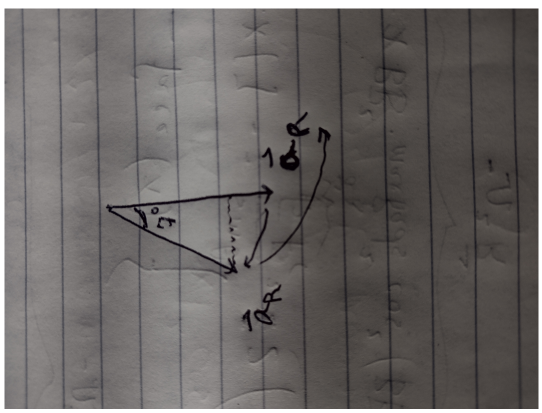

Figure 2: Gravity Vector

$\theta(t)$ initially increases, which means the reactor initially rotates counterclockwise and thus $\boldsymbol{a}_R$ rotates clockwise, and begins vertical. Thus the x component of $\boldsymbol{a}_R$ should be a sine function of θ, and the y component should be a cosine function, and both compoents should be negative since the motion is counterclockwise initially, and the starting direction is downwards. So the following function for $\boldsymbol{a}_R(t)$ is appropriate:

$$\boldsymbol{a}_R(t) = -\frac{sin\theta}{\sqrt{2}Fr^2}\hat{\mathbf{i}} - \frac{cos\theta}{\sqrt{2}Fr^2}\hat{\mathbf{j}} \tag{4}$$

This gives $|\boldsymbol{a}_R| = \frac{1}{Fr^2}$, where Fr is the Froude number, which matches with my nondimensionalization of the Navier-Stokes equation.

Next I consider the Coriolis acceleration, $2\boldsymbol{\Omega} \times \boldsymbol{u}_R$. Where $\boldsymbol{u}_R = u_x\hat{\mathbf{i}} + u_y\hat{\mathbf{j}} + 0\hat{\mathbf{k}} = \langle u_x, u_y, 0\rangle$, I can write $2\boldsymbol{\Omega} \times \boldsymbol{u}_R = 2\langle 0, 0, \frac{2\pi\Theta_{max}}{T} cos(\frac{2\pi}{T}t)\rangle \times \langle u_x, u_y, 0\rangle = 2\langle -u_y\frac{2\pi\Theta_{max}}{T} cos(\frac{2\pi}{T}t), u_x\frac{2\pi\Theta_{max}}{T} cos(\frac{2\pi}{T}t), 0\rangle = -2u_y\frac{2\pi\Theta_{max}}{T} cos(\frac{2\pi}{T}t)\hat{\mathbf{i}} + 2u_x\frac{2\pi\Theta_{max}}{T} cos(\frac{2\pi}{T}t)\hat{\mathbf{j}}$.

Finally I consider centripetal acceleration, $-\Omega^2\boldsymbol{R}$. The coordinate system is set up so that the center of the ellipse is $(0,0)$, and where l is the length of the semiminor axis, the pivot point is at $(0,-l)$, since the ellipse pivots about the point on its edge directly below the center, as illustrated in the following image from google images:

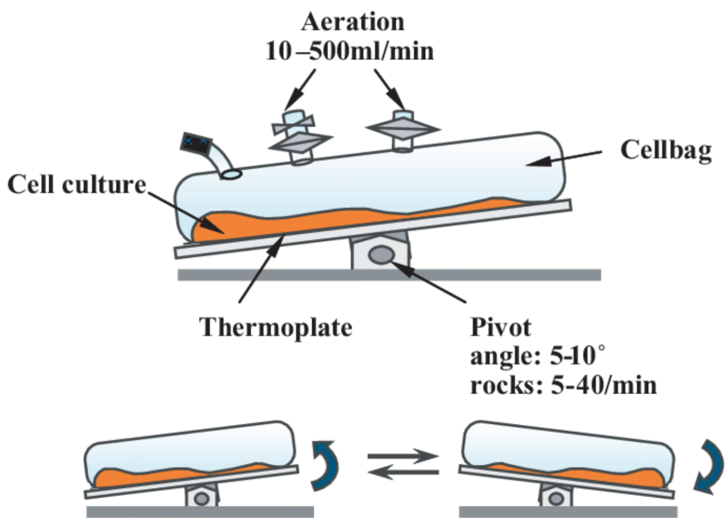

Figure 3: Reactor Pivot

Thus the position vector $\boldsymbol{R} = x\hat{\mathbf{i}} + (y+l)\hat{\mathbf{j}}$ for any point (x,y). This gives $-\Omega^2\boldsymbol{R} = -x\frac{4\pi^2\Theta_{max}^2}{T^2} cos^2(\frac{2\pi}{T})\hat{\mathbf{i}} - (y+l)\frac{4\pi^2\Theta_{max}^2}{T^2} cos^2(\frac{2\pi}{T})\hat{\mathbf{j}}$

3 Translation to Code

In Basilisk, I use foreach_face(x) and foreach_face(y) arguments in the acceleration event to handle the gravity, Coriolis and centripetal accelerations. I start by defining a few things at the beginning of the code, including Θ_{max}, called maxRads and converted from maxDegrees, and the period of oscillation, which is converted to a constant that appears in the cosine function:

```
double thetaNOW;
double omegaNOW;

const double period = 80.0; // how many seconds it
    takes to go through a
```

```
complete rocking cycle
double BB = (2.0*3.14159265)/period;

const double maxDegrees = 7.0; // degrees through
    which the reactor rotates
double maxRads = maxDegrees*(3.14159265/180.0);

// The following doubles will be used in the
    acceleration event to avoid really
long lines of code:
double gravityX;
double gravityY;
double coriolisX;
double coriolisY;
double centripetalX;
double centripetalY;
```

Then much later in the code, I use these values in the event setting up the acceleration term of the Navier-Stokes discretization:

```
event acceleration (i++)
{
  thetaNOW = maxRads*sin(BB*t); // Derivation in
      notebook. Should write this up at some point.
  omegaNOW = BB*maxRads*cos(BB*t); // Derivative of
      omegaNOW w/respect to t
  face vector av = a;
  foreach_face(x) {
    gravityX = (1/sqrt(2))*sq(1/Fr)*sin(thetaNOW);
    coriolisX = 2.0*((u.y[]+u.y[-1])/2.0)*BB*maxRads*
        cos(BB*t);
    centripetalX = x*sq(BB)*sq(maxRads)*sq(cos(BB*t))
        ;
    av.x[] -= gravityX + coriolisX + centripetalX;
  }
  foreach_face(y) {
    gravityY = (1/sqrt(2))*sq(1/Fr)*cos(thetaNOW);
    coriolisY = -2.0*((u.x[]+u.x[-1])/2.0)*BB*maxRads
        *cos(BB*t);
    centripetalY = sq(BB)*sq(maxRads)*sq(cos(BB*t))*(
        y+semiminor);
    av.y[] -= gravityY + coriolisY + centripetalY;
  }
}
```

As can be seen in the foreach arguments, the acceleration vector is defined using a "-=" symbol, which means the signs of the terms defined in the code are opposite those in the mathematical derivation. For example, the gravitational acceleration in the rotating reference frame is $\boldsymbol{a}_R(t) = -\frac{sin\theta}{\sqrt{2}Fr^2}\hat{\mathbf{i}} - \frac{cos\theta}{\sqrt{2}Fr^2}\hat{\mathbf{j}}$, but in the code I have gravityX and gravityY as positive. Likewise, the Coriolis term is $2\boldsymbol{\Omega} \times \boldsymbol{u}_R = -2u_y\frac{2\pi\Theta_{max}}{T}cos(\frac{2\pi}{T}t)\hat{\mathbf{i}} + 2u_x\frac{2\pi\Theta_{max}}{T}cos(\frac{2\pi}{T}t)\hat{\mathbf{j}}$, but in the code I have coriolisX as positive and coriolisY as negative. Finally, the centripetal term is $-\Omega^2\boldsymbol{R} = -x\frac{4\pi^2\Theta_{max}^2}{T^2}cos^2(\frac{2\pi}{T})\hat{\mathbf{i}} - (y+l)\frac{4\pi^2\Theta_{max}^2}{T^2}cos^2(\frac{2\pi}{T})\hat{\mathbf{j}}$, but in the code I have centripetalX and centripetalY as both positive.

Additionally, it is clear in the code that for the Coriolis terms I did not simply use u.x[] and u.y[] as might be assumed from the mathematical derivation. Instead, I use the averages (u.x[]+u.x[-1])/2 and (u.y[]+u.y[-1])/2. This is because the acceleration vector is face-staggered but the velocities are centrally staggered, so I must approximate the velocity values at the cell faces in order for the setup to be consistent.

References

[1] Pijush K. Kundu, Ira M. Cohen, and David R. Dowling. *Fluid mechanics.* Academic Press, 2010. Equation 4.53.