

Like a Record, Baby: The Rotating Coordinate Transformation (Atmo 336, Fall 2007)

Getting Started. Life would be a lot easier if we lived in an inertial reference frame. (Ok, maybe just dynamics class would be easier.) But we don't: we live in a rotating frame. And that means our local coordinate system is continually being accelerated in towards the axis of the Earth. So what to do?

Well, in principle there are two ways to deal with this. The first is to simply avoid the issue altogether and to only deal with problems in a non-rotating inertial frame. We could then map the results back to our rotating frame whenever we needed a local description. But in practice this tends to be a bit clumsy. A better approach is to solve for the motion in the rotating frame directly but to add extra forces to our equations to account for the frame's acceleration.

Our goal here is to show that these two approaches are in fact the same. And hopefully we'll build some intuition along the way.

Getting the Files. Copy the scripts *initial.m*, *acceleration.m*, *integrate.m* and *coord_map.m* from the shared *ATMO336* directory (under *lab_5_scripts*). The first three scripts are for computing a particle trajectory given a specified initial condition for the particle as well as a specified force field. These scripts could apply to either the inertial or the rotating frames—the only difference is in how you specify the initial conditions and the accelerations (as addressed further below). The final script maps a trajectory computed in the inertial frame into the corresponding trajectory in the rotating frame.

We'll do this in two steps. In *Step 1* we'll set up our three trajectory scripts so as to calculate the trajectory of the fluid particle as seen from the inertial frame. To finish *Step 1* we'll then run *coord_map.m* to get the same trajectory as viewed from the rotating frame. In *Step 2* we'll then go back and modify our trajectory scripts so as to compute the trajectory in the rotating frame directly (i.e., without a coordinate map). And with any luck, the trajectory computed in *Step 2* will match that from *Step 1*.

Step 1: The Inertial Frame and Rotating Coordinate Map. In the inertial frame we can apply Newton's second law directly. Suppose we let the position of a fluid particle (in 2D) be denoted by $\mathbf{x} = (x, y)$ so that the velocity of the particle is

$$\begin{aligned}\mathbf{u} &= (u, v) \\ &= \frac{d\mathbf{x}}{dt} = \left(\frac{dx}{dt}, \frac{dy}{dt} \right)\end{aligned}\tag{1}$$

Suppose further that the net force/mass on the particle is given by $\mathbf{F} = (F_x, F_y)$, where for our purposes we take F_x and F_y to be known functions of x , y and t . Then according to our main man Newton we have

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}\tag{2}$$

and separating (1) and (2) into components gives us the four equations

$$\frac{du}{dt} = F_x(x, y, t) \quad \frac{dv}{dt} = F_y(x, y, t) \quad (3)$$

$$\frac{dx}{dt} = u \quad \frac{dy}{dt} = v \quad (4)$$

where again we assume that F_x and F_y are known functions.

Now to get a trajectory we'll certainly need to integrate (4). But before we can integrate (4) we first need to know u and v , which means we also need to solve (3). But of course F_x and F_y depend on x and y , so to solve (3) we first need to solve (4). And of course the solution to (4) depends on u and v so that.....dizzy yet?

All this really means is that the four equations in (3) and (4) are coupled (meaning that the rate of change for one depends on the solution from the other) and must therefore be solved *simultaneously*. Luckily, your friendly neighborhood TA (FNNTA) has already written a script (the *integrate.m* script) to do this for you. So for the moment, all you need to know is that to make the script work you'll need to provide the initial values for x , y , u and v (at time $t = 0$) as well as the definitions for the forcing functions F_x and F_y . Hence the scripts *initial.m* and *acceleration.m*.

The initial conditions. When you first run the *integrate* script you'll be asked to input (at the matlab command line) the initial values for x , y , u and v as viewed from the inertial frame. This information is then sent to the *initial.m* script where it is used to set up the initial state for your trajectory calculation.

Wait a minute, that's strange. If we've already read the initial conditions from the prompt, then why do we need the script? The reason is that when you get to the rotating case in *Step 2* your initial conditions in the rotating frame will be different from those in the inertial frame (as you'll show in the next homework). So you'll have to map the inertial values entered at the prompt into the corresponding rotating values needed for the integration. And hence the script. But for now everything is in the inertial frame, so you don't have to do anything to *initial.m* except take a look.

The forcing functions. The forcing functions are specified in *acceleration.m*. Note in particular that *acceleration.m* requires two sets of forces. The first set is for F_x and F_y as described above. For the moment you can set these two to zero, since that's the first case you'll need to run below. The second set of forcing definitions is for the *apparent* forces. But the apparent forces are only relevant in the rotating case, so for now you should set these lines to zero also.

The time integration. Once you have *initial.m* and *acceleration.m* set up, then running the *integrate.m* script will produce a trajectory. Give it a try. Put in various initial conditions at the prompt and see if the trajectories match your expectations. (But be careful to keep your initial conditions in the plot domain—namely $-2000 \leq x \leq 2000$ and $-2000 \leq y \leq 2000$ —or else you won't see anything in the plot.) And if you're really adventurous you can also play around with the F_x and F_y definitions, just to see how that changes things. But don't waste

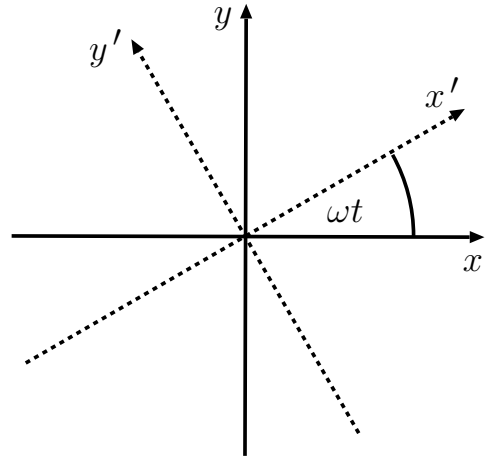
too much time—there’s plenty more to do below.

For reference, the *integrate.m* script is completely self-contained and does not need to be modified for either step. But feel free to prop up the hood and take a look. The time integration is a simple 2nd-order Runge-Kutta scheme (namely the midpoint method), for those of you who’ve had a numerics course.

The coordinate map. Once you’ve run *integrate.m* and have a trajectory, then running *coord_map.m* will map this trajectory to the rotating frame. The details of the coordinate mapping will be derived in the next homework. To be clear, we assume that the rotating coordinates are denoted by (x', y') and that the rate of rotation (in terms of angular frequency) is given by ω . We also assume that at time $t = 0$ the two coordinate systems coincide. Then the mapping between the stationary coordinates (x, y) and the rotating position (x', y') is given by (see problem 1 of the homework)

$$\begin{aligned}x' &= x \cos(\omega t) + y \sin(\omega t) \\y' &= -x \sin(\omega t) + y \cos(\omega t)\end{aligned}\tag{5}$$

But of course the formula is missing from *coord_map.m*, so you’ll have to add it. (Note that *coord_map.m* knows about the variable *omega*—it was defined when you ran *integrate.m*.)



Cases to run. Take a look at the following cases:

(a) First we’ll consider the case in which the particle is stationary in the inertial frame. You’ll also do this one as a homework problem, so try to remember what the answer looks like.

A stationary particle means no net force, so we’ll set

$$F_x = F_y = 0$$

and for the initial conditions (entered at the prompt) we’ll use

$$x(t = 0) = 0, \quad y(t = 0) = 1000, \quad u(t = 0) = 0, \quad v(t = 0) = 0$$

Run *integrate.m* and save your movie to *inertial_1.avi* [using *movie2avi* ($M, 'inertial_1'$)]. Then run *coord_map.m* and save the result to *map_1.avi*. Think about the result: is it what you expected? If not, go back and try, try again.

(b) Next we’ll look at an example done in class. Specifically, we’ll consider the case of straight-line motion in the y -direction with no forces applied.

Again no forces applied so leave

$$F_x = F_y = 0$$

and for the initial conditions (entered at the prompt) we’ll use

$$x(t = 0) = 0, \quad y(t = 0) = 0, \quad u(t = 0) = 0, \quad v(t = 0) = 20$$

Save your inertial movie to *inertial_2.avi* and your coordinate-mapped version to *map_2.avi*. Does the inertial movie make sense given the absence of forces in this case? And as for the coordinate-mapped movie—does the particle bend in the right direction given the sense of rotation?

(c) And for our final example we'll consider simple harmonic motion about the origin. That is, the spring problem—with the resting point for the spring being the origin.

To get a harmonic oscillation we'll need a restoring force that's proportional to the displacement. In vector form this looks like

$$\mathbf{F} = -\kappa\mathbf{x} \tag{6}$$

where for the present case we'll use $\kappa = 1/250$. For initial conditions we'll try

$$x(t = 0) = 0, \quad y(t = 0) = 1000, \quad u(t = 0) = 0, \quad v(t = 0) = 0$$

Again save your inertial movie to *inertial_3.avi* and your coordinate-mapped version to *map_3.avi*. Does the inertial movie make sense given the forces here? And I'll bet you didn't guess the coordinate-mapped answer beforehand.

Step 2: Adding the Apparent Forces. Ok, so we've computed the rotating trajectory by first computing the inertial trajectory and then mapping the coordinates. But as mentioned previously, this is typically a clumsy way to go. So now we'll step back and take the alternative approach—that is, we'll compute the trajectory in the rotating frame from the start. But to do this we'll first have to add some extra forces.

We'll let the position in the rotating frame be given by

$$\mathbf{x}' = x' \hat{\mathbf{x}}' + y' \hat{\mathbf{y}}' = (x', y') \tag{7}$$

where $\hat{\mathbf{x}}'$ and $\hat{\mathbf{y}}'$ of course refer to unit vectors along the x' and y' axes. Note that the two vectors \mathbf{x} and \mathbf{x}' refer to the same physical quantity—the position of the particle—and the two are thus in some sense the same thing. We're just measuring this quantity relative to two different coordinate axes. In any case, having defined the position by (7) the corresponding velocity in the rotating frame must be

$$\mathbf{u}' = (u', v') = \left(\frac{dx'}{dt}, \frac{dy'}{dt} \right) \tag{8}$$

Now as mentioned previously the rotating coordinates are *non-inertial*, meaning that a stationary point as viewed from the rotating frame is actually continuously accelerated as viewed in the inertial frame. So we can't just apply Newton's 2nd law directly like we did before. Instead, we first need to throw in a couple of *apparent* forces to account for the frame's acceleration. In fact, as shown in class we need two apparent forces: the outward-directed *centrifugal* force

$$\mathbf{a}_{\text{cent}} = \omega^2 \mathbf{x}' \tag{9}$$

and the Coriolis force

$$\mathbf{a}_{\text{cor}} = -2\boldsymbol{\omega} \times \mathbf{u}' = 2\omega(v', -u') \quad (10)$$

The *angular velocity vector* $\boldsymbol{\omega}$ in (10) is defined to be a vector of length ω directed along the rotation axis. In this case that means $\boldsymbol{\omega} = \omega \hat{\mathbf{z}}$.

As before we let the real force on the particle be given by the force vector $\mathbf{F} = (F_x, F_y)$, which we now take to be a function of x' and y' . To get our momentum equations in the rotating frame we then combine this real force with our two apparent forces to get

$$\frac{d\mathbf{u}'}{dt} = \mathbf{F} + \omega^2 \mathbf{x}' - 2\boldsymbol{\omega} \times \mathbf{u}'$$

which after separating into components and combining with (8) gives the four component equations

$$\frac{du'}{dt} = F_x(x', y', t) + \omega^2 x' + 2\omega v' \quad \frac{dv'}{dt} = F_y(x', y', t) + \omega^2 y' - 2\omega u' \quad (11)$$

$$\frac{dx'}{dt} = u' \quad \frac{dy'}{dt} = v' \quad (12)$$

Given the initial values for x' , y' , u' and v' , we can then integrate (11) and (12) (using *integrate.m* as before) to get the trajectory directly in the rotating frame—without ever having to do a coordinate transformation.

The initial conditions. The values you enter at the Matlab prompt are actually the initial values as seen from the inertial frame. So to get the initial values in the rotating frame you'll first have to apply a transformation. Again, the details will be derived in the homework. But the end result is that the coordinate transformation is again (5) while the corresponding velocity transformation is

$$\begin{aligned} u' &= (u + \omega y) \cos(\omega t) + (v - \omega x) \sin(\omega t) \\ v' &= (-u - \omega y) \sin(\omega t) + (v - \omega x) \cos(\omega t) \end{aligned}$$

So to get the initial conditions you just need to evaluate these formulae at $t = 0$. And then code your results into *initial.m*.

The accelerations. The real forces F_x and F_y haven't changed, so you can leave these two lines as before. But now you need the two apparent forces as well. So guess what gets added as *dudt_app* and *dvdt_app*?

The time integrations. Once you've got your initial state and accelerations set up, then running *integrate.m* will compute the trajectory just as before. But of course now you're using the rotating equations (11) and (12)—as well as the rotating initial conditions—so the trajectory computed by *integrate.m* will then be that seen from the rotating frame.

Cases to run. We'll consider the same three cases that we looked at previously:

(d) First the stationary particle in the inertial frame. As before, the forces and initial state for this problem are

$$F_x = F_y = 0$$

and

$$x(t = 0) = 0, \quad y(t = 0) = 1000, \quad u(t = 0) = 0, \quad v(t = 0) = 0$$

where again the initial values described here refer to those seen in the inertial frame (and then entered at the Matlab prompt). Run *integrate.m* and save your result to *rotating_1.avi*. Does your trajectory match the one computed in *Step 1*? If not, go back and try, try again.

(e) Now for the straightline motion case. The forces for this one are again zero, while the initial state is given by

$$x(t = 0) = 0, \quad y(t = 0) = 0, \quad u(t = 0) = 0, \quad v(t = 0) = 20$$

Run *integrate.m* and save your result to *rotating_2.avi*. Does your trajectory match the one computed in *Step 1*?

(f) And finally the harmonic oscillation. The force vector for this case was given in the inertial coordinates by (6). But if we recall that \mathbf{x} and \mathbf{x}' refer to the same physical quantity—just measured relative to different axes—then the corresponding force as measured relative to the rotating axes must be

$$\mathbf{F} = -\kappa\mathbf{x}'$$

(Convince yourself this is correct.) And as before the initial conditions are

$$x(t = 0) = 0, \quad y(t = 0) = 1000, \quad u(t = 0) = 0, \quad v(t = 0) = 0$$

Run *integrate.m* and save your result to *rotating_3.avi*. How does this compare to your previous calculation?

(g) If you have time, make up some additional cases. (If you come up with an interesting one it may find its way into a future lab. Be creative—this could be your legacy!) And see if you can predict the results beforehand.

Movie Checklist. When finished you should have turned in the following movies:

inertial_1.avi, map_1.avi, inertial_2.avi, map_2.avi, inertial_3.avi, map_3.avi
rotating_1.avi, rotating_2.avi, rotating_3.avi

As before, turn to the student at your right and argue over whose movies are best.