Direction Cosine Matrix IMU: Theory

William Premerlani and Paul Bizard

This is the first of a pair of papers on the theory and implementation of a direction-cosine-matrix (DCM) based inertial measurement unit for application in model planes and helicopters. Actually, at this point, it is still a draft, there is still a lot more work to be done. Several reviewers, especially Louis LeGrand and UFO-man, have made good suggestions on additions and revisions that we should make and prepared some figures that we have not included yet. We will eventually incorporate their suggestions, but it may take a long time to get there. In the meantime, we think there is an audience who can benefit from what we have so far.

The motivation for DCM was to take the next step in stabilization and control functions from an inherently stable aircraft with elevator and rudder control, to an aerobatic aircraft with ailerons and elevator. One of the authors (Premerlani) built a two axes board several years ago, and developed rudimentary firmware to provide stabilization and return-to-launch (RTL) functions for a Gentle Lady sailplane. The firmware worked well enough, and the author came to rely on the RTL feature, but it never seemed to work as well as the author would like. In particular, satisfactory solutions to the following two issues were never found:

- Mixing. It was recognized that in a banked turn, there were two problems arising from the bank angle. First, the yaw rotation of the aircraft around the turn generated a nuisance signal in the pitch gyro, because of the banking. Second, in order to make a level turn, the elevator needed some “up” deflection. The amount of deflection depends on the bank angle, which could not be directly measured. Both issues were opposite sides of the same coin.
- Acceleration. An accelerometer measures gravity minus acceleration. The acceleration is equal to the total of all of the aerodynamic forces (lift, thrust, drag, etc.) on the plane, plus the gravity force, divided by the mass. Therefore, the accelerometer measures the negative of the total of all of the aerodynamic forces. The measurement of gravity is what is needed to level the plane but that is not what you get out of an accelerometer during accelerated motion. Acceleration is a confounding variable. In particular, when the aircraft pitches up or down, for a short while it accelerates in such a way that the output of an accelerometer does not change. There is a similar effect that the NASA astronauts experience when they are in training planes. A ballistic path can produce zero net forces and therefore fool accelerometers temporarily. The combination of this issue and the previous one prevented really tight

pitch control, and this issue prevented the use of pitch stabilization during a hand launch.

It was realized that part of the problem was not having a six degree of freedom inertial measurement unit (IMU), so it was decided to design a new board. The UAV DevBoard from SparkFun was the result.

Coincidentally, one of us (Premerlani) decided he wanted to step up to an aircraft with ailerons, and found that he just did not have the needed flying skills. He crashed 5 times in one summer, and had to completely replace his plane 3 times. So, he decided to use his new board for stabilization, shown below, attached to his Goldberg Endurance with Velcro.

The question was, how best to do that? Working together, we came to the same conclusion of Mahoney [1]. What is needed is a method that “fully respects the nonlinearity of the rotation group.” Paul and I decided that we should represent the rotation with a direction cosine matrix, that we could maintain the elements of the matrix using gyro, accelerometer, and GPS information, and that we could use the matrix for control and navigation. At a high level, here is how DCM works:

1. The gyros are used as the primary source of orientation information. We integrate the nonlinear differential kinematic equation that relates the time rate of change in the orientation of the aircraft to its rotation rate, and its present orientation. This is done at a high rate, (40 to 50 Hz) often enough to give the servos fresh information for each and every PWM pulse that is sent to the servos.

2. Recognizing that numerical errors in the integration will gradually violate the orthogonality constraints that the DCM must satisfy, we make regular, small adjustments to the elements of the matrix to satisfy the constraints.
3. Recognizing that numerical errors, gyro drift, and gyro offset will gradually accumulate errors in the DCM elements, we use reference vectors to detect the errors, and a proportional plus integral (PI) negative feedback controller between the detected errors and the gyro inputs used in step 1, to dissipate the errors faster than they can build up. GPS is used to detect yaw error, accelerometers are used to detect pitch and roll.

The process is shown schematically in Figure 1.

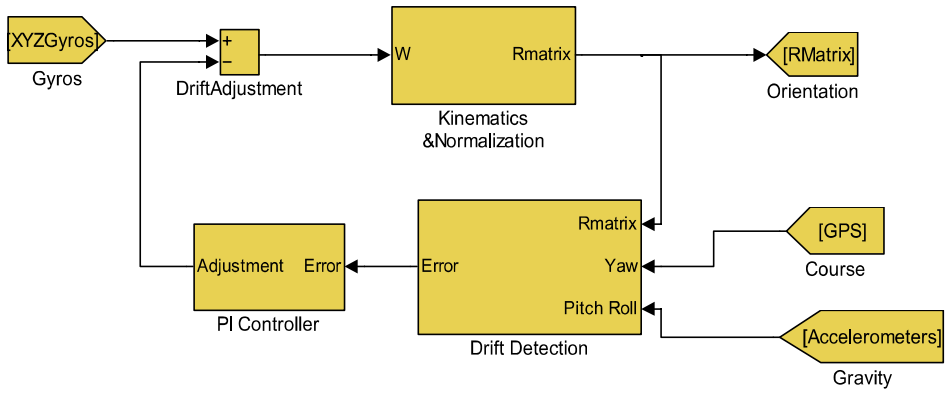


Figure 1 Block diagram of DCM

No doubt you are wondering what a rotation group is, and why it should be respected. You also might be wondering how you can use DCM for control and navigation. You also might have the same questions that UFO-MAN asked on the subject after he read Mahoney's paper, so we will start with those questions:

- What is a quaternion and why do we use that instead of vector notation?
- What is meant by a rotation group?
- What is a rotation matrix?
- What does it mean to maintain orthogonality of the rotation matrix?
- What is an anti symmetric matrix?
- Can you briefly explain kinematics in this rotation matrix context?
- Can you briefly explain dynamics in this rotation matrix context?

All of this is concerned with rotations. Physically, what we are trying to do is represent the orientation of our aircraft with respect to the earth as a rotation. There are several ways to do this. Mahony's paper discusses two alternate ways, rotation matrices and quaternions. Both approaches are similar in motivation, they are representing rotations without approximations and without singularities. Quaternions have the advantage of requiring only 4 values, while rotation matrices have 9. Rotation matrices have the advantage of being a natural fit to control and navigation. We chose rotation matrices as having a slight advantage, and for being more familiar to us.

$$R = \begin{array}{c|ccc|} & xb & yb & zb \\ \hline xe & & & \\ ye & & & \\ ze & & & \\ \hline \end{array} \qquad R^{-1} = \begin{array}{c|ccc|} & xe & ye & ze \\ \hline xb & & & \\ yb & & & \\ zb & & & \\ \hline \end{array}$$

A rotation matrix describes the orientation of one coordinate system with respect to another. The columns of the matrix are the unit vectors in one system as seen in the other system. A vector in one system can be transformed into the other system by multiplying it by the rotation matrix. The transformation in the reverse direction is accomplished with the inverse of the rotation matrix, which turns out to be equal to its transpose. (The transpose is just the swap of rows and columns.) Unit vectors are useful in the control and navigation computations, because their length is one. Therefore they can be used in dot and cross products to obtain the sine or cosine of various angles.

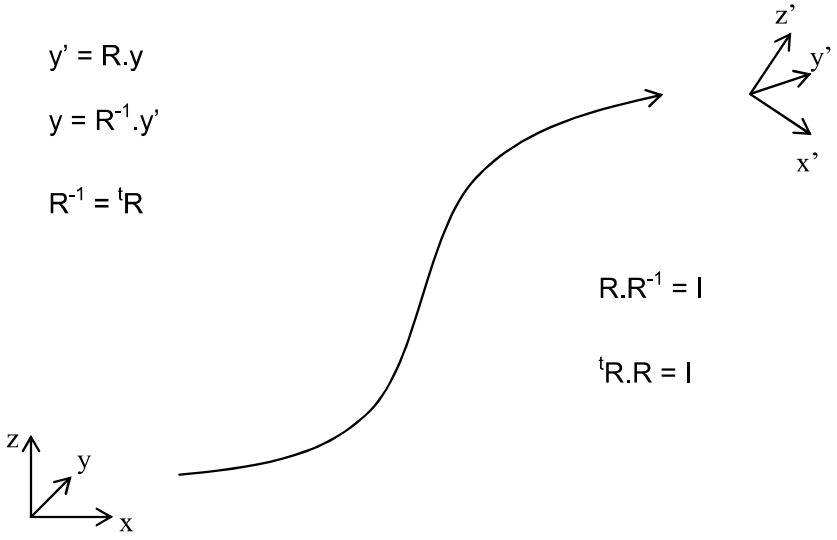
$y' = R.y$

$y = R^{-1}.y'$

$R^{-1} = {}^{t}R$

$R.R^{-1} = I$

${}^{t}R.R = I$



As your plane flies along, it is possible to describe its motion with a translation (movement of its center of gravity) and a rotation (change in orientation around its center of gravity). Its orientation with respect to the earth can be described by specifying a rotation about an axis. By starting with the plane level and pointing in a standard direction and applying the rotation, you will place the plane in its actual orientation. Any orientation can be described as a rotation from the “standard” position.

A rotation group is the group of all possible rotations. It is called a group because any two rotations in the group can be composed to produce another rotation in the group, every rotation has an inverse rotation, and there is an identity rotation. That is the definition. However, the way that we like to think about it as being a group is that you can wind up going around a complete circle and arriving back where you started. The rotation group is closed.

The reason that the rotation group should be respected is that by doing that, you make the fewest approximations and are able to perform control and navigation with the plane in any orientation, including upside down and pointing vertical. You can do aerobatics without making any approximations.

The basic idea is that the rotation matrix that defines the orientation of your aircraft can be maintained by integrating the nonlinear differential equation that describes the kinematics of the rotation. (We will present the nonlinear differential equation shortly, and explain why it is nonlinear.) Kinematics is concerned with the geometry of the rotation of a rigid body, and how the rotation transforms one rigid configuration into another configuration. This is done by recognizing that the integration can be accomplished via a series of matrix compositions.

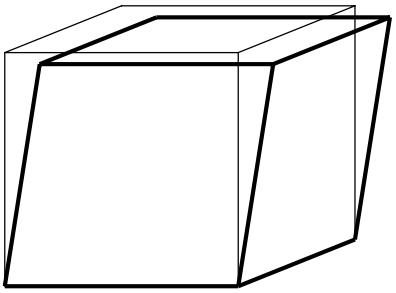
By matrix composition we simply mean multiplying two rotation matrices together. It can be shown that the resulting matrix represents the net rotation that results from applying the two rotations in sequence that each of the matrices represents.

However, numerical integration introduces numerical errors, and does not produce the same result that symbolic integration. An exact symbolic integration of the exact gyro signals will produce the exactly correct rotation matrix. Numerical integration, even if we had the exact gyro signals, will introduce two sorts of numerical errors:

- Integration error. Numerical integration uses a finite time step and data that is sampled at a finite sampling rate. Depending on the method that you use to do the integration, you are making certain assumptions about what is happening between data samples. The method that we use in our implementation assumes that the rotation rate is constant over the time step. This introduces an error that is proportional to the rotational acceleration.
- Quantization error. No matter what representation you use for the values, the digital representation is finite, so there is a quantization

error, starting at the analog to digital converter, and building whenever you perform a calculation that does not preserve all of the bits of the result.

One of the key properties of the rotation matrix is its orthogonality, which means that if two vectors are perpendicular in one frame of reference, they are perpendicular in every frame of reference. Also, that the length of a vector is the same in every frame of reference. Numerical errors can violate this property. For example, since the rows and columns are supposed to represent unit vectors, their magnitude should be equal to one, but numerical errors could cause them to get smaller or larger. Eventually they could shrink to zero, or go to infinity. The rows and columns are supposed to be perpendicular to each other, numerical errors could cause them to "lean" into each other, as shown below:



The rotation matrix has 9 elements. Actually, only 3 of them are independent. The orthogonality property of the rotation matrix in mathematical terms means that any pair of columns (or rows) of the matrix are perpendicular, and that the sum of the squares of the elements in each column (or row) is equal to 1. So, there are 6 constraints on the 9 elements.

$$||xb|| = 1$$

$$||yb|| = 1$$

$$||zb|| = 1$$

$$xb \perp yb$$

$$xb \perp zb$$

$$yb \perp zb$$

$$R = \begin{array}{c|ccc|} & xb & yb & zb & \\ \hline & & & & xe \\ & & & & ye \\ & & & & ze \\ \hline \end{array}$$

An antisymmetric matrix is a matrix in which each element in the matrix is equal to the negative of the element with swapped row and column index. So, for example, if the element in the first row, third column is 0.5, then the

element in the third row, first column must be -0.5. Also, the elements on the diagonal of an antisymmetric matrix must be zero.

It turns out that a small rotation can be described with an antisymmetric matrix as shown below:

$$\begin{vmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{vmatrix}$$

In our case, kinematics is concerned with the implications of rigid body rotation. It results in a nonlinear differential equation that describes the time evolution of the orientation of the body in terms of its vector rotation rate. The direction cosine matrix is all about kinematics.

Dynamics in our case is the application of Newton's laws to describe the time rate of change of the rotation rate vector in terms of the applied torques.

By the way, the dynamics in Mahony's paper are NOT accurate for planes, they were concerned mainly with helis and vertical take-offs. Mahony's paper describes how to implement a combined orientation measurement and control algorithm. What Paul and I are doing involves kinematics only. We have completely ignored dynamics for now. The kinematics (rotation matrix) by itself is very useful for providing a basis for control and navigation of model airplanes.

You are probably still wondering how to use DCM. Control and navigation can be accomplished with DCM entirely in Cartesian coordinates using vector cross products and dot products. For example, at a high level, here is how you accomplish these four control and navigation calculations.

1. To control the pitch of an aircraft, you need to know the pitch attitude of the aircraft, which you can find by taking the dot product of the roll axis of the aircraft with the ground vertical.
2. To control the roll of an aircraft, you need to know the bank attitude of the aircraft, which you can find by taking the dot product of the pitch axis of the aircraft with the ground vertical.
3. To navigate, you need to know the yaw attitude of the aircraft with respect to the direction that you want to go, which you can find by taking the cross product of the roll axis of the aircraft with a vector in the direction that you want to go. This works even if you are upside down. To find out if your aircraft might be pointing in the opposite

direction than you want to go, take the dot product of the roll axis with the desired direction vector. If it is negative, the aircraft is more than 90 degrees off course.

4. To find out if the aircraft is upside down, examine the sign of the dot product of the aircraft yaw axis with the vertical. If it is less than zero, the aircraft is upside down.
5. To find out the turning rate of the aircraft around the vertical earth axis, transform the gyro rotation vector to the earth frame of reference, and take the dot product with the vertical axis.

We now get deeper into the details of the theory.

Axis conventions

To describe the motion of an airplane it is necessary to define a suitable coordinate system. For most problems dealing with aircraft motion, two coordinate systems are used. One coordinate system is fixed to the earth and may be considered for the purpose of aircraft motion analysis to be an inertial coordinate system. The other coordinate system is fixed to the airplane and is referred to as a body coordinate system. Figure 2 shows the two right-handed coordinate systems.

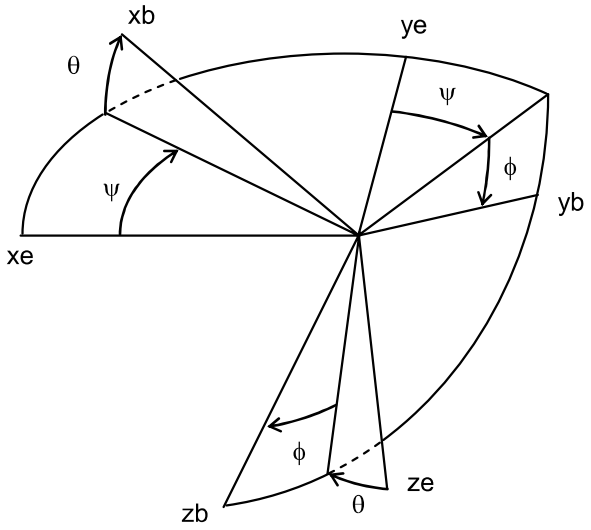


Figure 2 Body fixed frame and earth fixed frame

The orientation of the airplane is often described by three consecutive rotations, whose order is important. The angular rotations are called the Euler angles. The orientation of the body frame with respect to the fixed earth frame can be determined in the following manner. Imagine the airplane to be

positioned so that the body axis system is parallel to the fixed frame and then apply the following rotations:

1. Rotate the body about its zb axis through the yaw angle ψ
2. Rotate the body about its yb axis through the pitch angle θ
3. Rotate the body about its xb axis through the roll angle ϕ

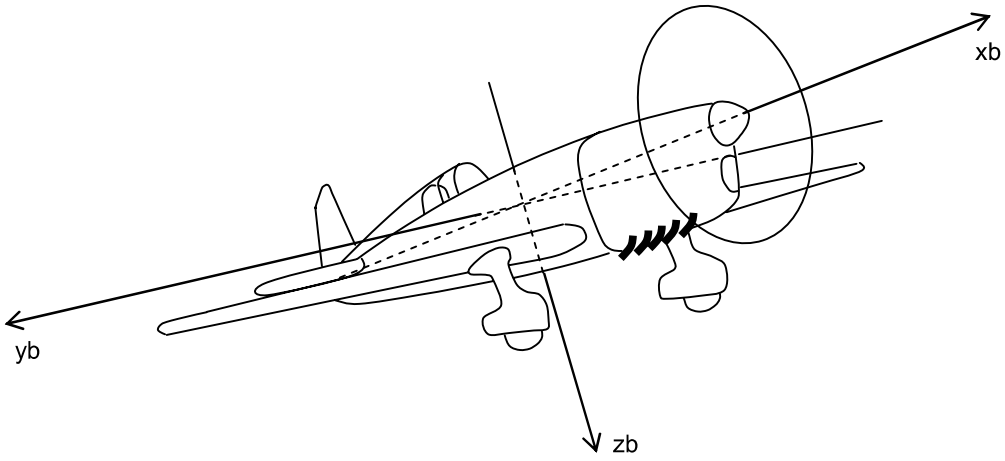


Figure 3 Body axes coordinate system

Direction cosine matrices

Certain types of vectors, such as directions, velocities, accelerations, and translations, (movements) can be transformed between rotated reference frames with a 3X3 matrix. We are interested in the plane frame of reference and the ground frame of reference. It is possible to rotate vectors by multiplying them by a matrix of direction cosines:

$$\mathbf{Q} = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \end{bmatrix} = \text{a vector, such as a direction, velocity or acceleration}$$

$$\mathbf{Q_P} = \text{a vector } \mathbf{Q} \text{ measured in the frame of reference of the plane}$$

$$\mathbf{Q_G} = \text{a vector } \mathbf{Q} \text{ measured in the frame of reference of the ground} \qquad \text{Eqn. 1}$$

$$\mathbf{R} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix} = \text{rotation matrix}$$

$$\mathbf{Q_G} = \mathbf{R}\mathbf{Q_P}$$

The relation between the direction cosine matrix and Euler angles is:

$$\mathbf{R} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \quad \text{Eqn. 2}$$

Equation 1 and equation 2 expresse how to rotate a vector measured in the frame of reference of the plane to the frame of reference of the ground. Equation 1 is expressed in terms of direction cosines. Equation 2 is expressed in terms of Euler angles.

In equation 1, each component of the vector in the ground frame is equal to the dot product of the corresponding row of the rotation matrix with the vector in the plane frame. Nine multiplies and six additions are required to compute the rotation. Equation 3 is a restatement of equation 1, with the matrix multiplication expanded in terms of the elements of the vectors and the matrix.

$$\begin{aligned} Q_{Gx} &= r_{xx}Q_{Px} + r_{xy}Q_{Py} + r_{xz}Q_{Pz} \\ Q_{Gy} &= r_{yx}Q_{Px} + r_{yy}Q_{Py} + r_{yz}Q_{Pz} \\ Q_{Gz} &= r_{zx}Q_{Px} + r_{zy}Q_{Py} + r_{zz}Q_{Pz} \end{aligned} \quad \text{Eqn. 3}$$

Note that the R matrix is not necessarily symmetric. The three columns of the R matrix are the transformations of the three axis vectors of the plane to the ground frame of reference. The three rows of the R matrix are the transformations of the three axis vectors of the ground coordinate system to the plane frame of reference. The R matrix contains all the information needed to express the orientation of the plane with respect to the ground. The R matrix is also called the direction cosine matrix, because each entry is the cosine of the angle between an axis of the plane and an axis on the ground. Although it would appear that there are 9 independent parameters in the R matrix, there are really only 3 independent ones, because of the six so-called orthogonality (also known as normalization) conditions: the three column vectors are mutually perpendicular and the magnitude of each column vector is equal to one.

The transpose of any matrix, and the rotation matrix in particular, indicated as $\mathbf{R}^{\mathrm{T}}$, is formed by interchanging rows and columns. In general, the inverse of a square matrix, if it exists, is indicated as $\mathbf{R}^{-1}$. The inverse of a matrix times the matrix produces the identity matrix. (The identity matrix has all ones on the diagonal, and all zeros everywhere else. Multiplying any matrix by the identity matrix leaves it unchanged. In the case of rotation matrices, it turns out that the transpose of the R matrix is equal to its inverse:

$$\mathbf{R}^{-1} = \mathbf{R}^{\mathrm{T}} = \begin{bmatrix} r_{xx} & r_{yx} & r_{zx} \\ r_{xy} & r_{yy} & r_{zy} \\ r_{xz} & r_{yz} & r_{zz} \end{bmatrix} \qquad \text{Eqn. 4}$$

$$\mathbf{Q}_{\mathbf{P}} = \mathbf{R}^{-1}\mathbf{Q}_{\mathbf{G}} = \mathbf{R}^{\mathrm{T}}\mathbf{Q}_{\mathbf{G}}$$

The reason that the inverse of the rotation matrix is equal to its transpose is because of the symmetry of the situation. The elements of the rotation matrix are the cosines between pairs of axis, one in the plane frame, and one in the ground frame. The inverse situation is equivalent to exchanging the roles of the ground and plane frame of reference, which is the same as interchanging rows and columns, which is the same as the transpose.

Also, the fact that the inverse is equal to the transpose is consistent with the orthogonality conditions, which can be expressed in matrix notation as:

$$\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{Eqn. 5}$$

Equation 5 can be used to prove that the inverse of R is R transpose, by multiplying the equation by the inverse of R, or by the inverse of R transpose.

A very useful property of the rotation matrix is that we can compose rotations. We can multiply several rotations matrices together, and get a rotation matrix that is equivalent to applying all of the rotations in succession. We have to be careful to apply the rotations in succession on the left side of what we already have. For example, if we have three rotation matrices, from orientation A to orientation B, from B to C, and from C to D, we can compute the rotation matrix that will go from orientation A to orientation D according to:

$$\begin{aligned} \mathbf{R}_{\mathbf{DA}} &= \mathbf{R}_{\mathbf{DC}}\mathbf{R}_{\mathbf{CB}}\mathbf{R}_{\mathbf{BA}} \\ \mathbf{R}_{\mathbf{BA}} &= \text{rotation matrix from A to B} \\ \mathbf{R}_{\mathbf{CB}} &= \text{rotation matrix from B to C} \\ \mathbf{R}_{\mathbf{DC}} &= \text{rotation matrix from C to D} \\ \mathbf{R}_{\mathbf{DA}} &= \text{rotation matrix from A to D} \end{aligned} \qquad \text{Eqn. 6}$$

The reason that we have to be careful about the sequence of operations when multiplying rotations matrices is that matrix multiplication is NOT commutative. That is, the order of matrix multiplication matters very much.. This is consistent with rotations, which are not commutative either. For example, consider what happens if a plane pitches around its own pitch and roll axes by 90 degrees each. The order very much matters. Suppose that is pitches up by 90 degrees, followed by a roll of 90 degrees. At that point the plane will be traveling vertically. However, if it rolls first, and then banks, it will be traveling in the horizontal plane.

Finally, there is a useful identity that applies to matrices in general, and to rotation matrices in particular. The transpose of the product of two matrices is equal to the product of the transposes of the matrices, with the two matrices swapped:

$$\left(\mathbf{AB}\right)^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} \qquad \mathbf{A},\mathbf{B} \text{ are matrices} \qquad \text{Eqn. 7}$$

Vector dot and cross products

Two very useful vector products that we will use in computing DCM and in using its elements for navigation and control are the dot product and the cross product. The dot product of two vectors A and B, is a scalar computed by performing a matrix multiplication of a A as a row vector with B as a column vector producing:

$$\mathbf{A}\cdot\mathbf{B} = \mathbf{A}^{\mathrm{T}}\mathbf{B} = \begin{bmatrix} A_x & A_y & A_z \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = A_xB_x + A_yB_y + A_zB_z \qquad \text{Eqn. 8}$$

It turns out that the vector dot product produces a result that is equal to the product of the magnitudes of the two vectors, times the cosine of the angle between them:

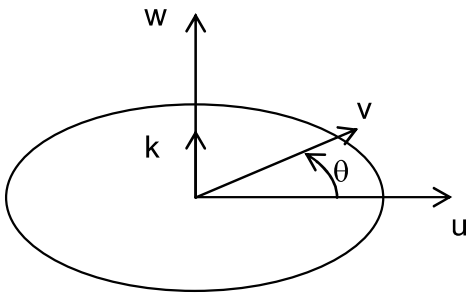
$$\mathbf{A}\cdot\mathbf{B} = \left|\mathbf{A}\right|\left|\mathbf{B}\right|\cdot\cos\left(\theta_{AB}\right) \qquad \text{Eqn. 9}$$

We note that the dot product is commutative: $\mathbf{A}\cdot\mathbf{B} = \mathbf{B}\cdot\mathbf{A}$

The cross product of two vectors A and B, is a vector whose components are computed by:

$$\begin{aligned} \left(\mathbf{A}\times\mathbf{B}\right)_x &= A_yB_z - A_zB_y \\ \left(\mathbf{A}\times\mathbf{B}\right)_y &= A_zB_x - A_xB_z \\ \left(\mathbf{A}\times\mathbf{B}\right)_z &= A_xB_y - A_yB_x \end{aligned} \qquad \text{Eqn. 10}$$

The cross product is perpendicular to both of its vector factors and its magnitude is proportional to the magnitudes of the vectors times the sine of the angle between them:



w = u ∧ v = ||u||.||v||.sin(θ).k

||w|| = ||u||.||v||.sin(θ)

k : unit vector orthogonal to the plane defined by u and v

Stated another way:

$$\begin{aligned}&(\mathbf{A}\times\mathbf{B})\cdot\mathbf{A}=(\mathbf{A}\times\mathbf{B})\cdot\mathbf{B}=0\\&|\mathbf{A}\times\mathbf{B}|=|\mathbf{A}||\mathbf{B}|\sin(\theta_{AB})\end{aligned}$$ Eqn. 11

We note that the cross product is anti-commutative. $\mathbf{A}\times\mathbf{B}=-\mathbf{B}\times\mathbf{A}$

Computing direction cosines from gyro signals

With the preliminaries out of the way, we now move on to the central concept of the DCM algorithm: the nonlinear differential equation that relates the time rate of change of the direction cosines to the gyro signals. Our goal is to compute the direction cosines without making any approximations that violate the nonlinearity of the equations. For the moment, we assume that the gyro signals have no errors. Later on we will address the issue of gyro drift.

Unlike rotating mechanical gyros, which stay fixed in space while the aircraft rotates around them, electronic rate gyros rotate with the aircraft, producing signals proportional to the rotation rate. Since rotations do not commute, and the sequence of rotations matter, we cannot get by with simply integrating the gyro rate signals to get angles, that will not work. What we have to do is look to the kinematics of rotations to see what we need to do to get the correct answer.

A well known result of kinematics is that the rate of change of a rotating vector due to its rotation is given by:

$$\begin{aligned}&\frac{d\mathbf{r}(t)}{dt}=\boldsymbol{\omega}(t)\times\mathbf{r}(t)\\&\boldsymbol{\omega}(t)=\text{rotation rate vector}\end{aligned}$$ Eqn. 12

We make the following observations:

1. The differential equation is nonlinear. The rotation vector input is (cross) multiplied by the variable that we are trying to integrate. Therefore, any linear approach will be only an approximation.
2. Both vectors must be measured in the same reference frame.
3. Because the cross product is anticommutative, we could reverse the order and change the sign.

If we know the initial conditions and the time history of the rotation vector, we can numerically integrate equation 11 to track the rotating vector:

$$\begin{aligned}
&\mathbf{r}(t) = \mathbf{r}(0) + \int_0^t d\boldsymbol{\theta}(\tau) \times \mathbf{r}(\tau) \\
&d\boldsymbol{\theta}(\tau) = \boldsymbol{\omega}(\tau) d\tau \\
&\mathbf{r}(0) = \text{starting value of the vector} \\
&\int_0^t d\boldsymbol{\theta}(\tau) \times \mathbf{r}(\tau) = \text{change in the vector}
\end{aligned} \qquad \text{Eqn. 13}$$

Our strategy is going to be to apply equation 13 to the rows or the columns of the R matrix, treating them as rotating vectors.

The first snag that we run into is that the vectors that we want to track, and the rotation vector, are not measured in the same reference frame. Ideally, we would like to track the axes of the aircraft in the earth frame of reference, but the gyro measurements are made in the aircraft frame of reference. There is an easy solution to the issue by recognizing the symmetry in the rotation. In the frame of reference of the plane, the earth frame is rotating equal and opposite to the rotation of the plane in the earth frame. So we can track the earth axes as seen in the plane frame by flipping the sign of the gyro signals. As a matter of convenience, we can flip the sign back, and interchange the factors in the cross product:

$$\begin{aligned}
&\mathbf{r}_{earth}(t) = \mathbf{r}_{earth}(0) + \int_0^t \mathbf{r}_{earth}(\tau) \times d\boldsymbol{\theta}(\tau) \\
&d\boldsymbol{\theta}(\tau) = \boldsymbol{\omega}(\tau) d\tau \\
&\mathbf{r}_{earth}(t) = \text{one of the earth axes, as viewed from the plane}
\end{aligned} \qquad \text{Eqn. 14}$$

The vectors in equation 14 are the rows of the R matrix in equation 1. The next question is how to conveniently implement equation 14. We take the same matrix approach that Mahoney [1] uses. We start by going back to the differential form of equation 14:

$$\begin{aligned}
&\mathbf{r}_{earth}(t + dt) = \mathbf{r}_{earth}(t) + \mathbf{r}_{earth}(t) \times d\boldsymbol{\theta}(t) \\
&d\boldsymbol{\theta}(t) = \boldsymbol{\omega}(t) dt
\end{aligned} \qquad \text{Eqn. 15}$$

There is one more thing that we need to do, in anticipation of the drift cancellation that we will be doing later on. We need to add the correction rotation rate that comes out of the proportional plus integral drift

compensation feedback controller to the measurement that the gyros make, to produce our best estimate of the true rotation rate:

$$\begin{aligned}&\boldsymbol{\omega}(t) = \boldsymbol{\omega}_{\text{gyro}}(t) + \boldsymbol{\omega}_{\text{correction}}(t)\\&\boldsymbol{\omega}_{\text{gyro}}(t) = \text{three axis gyro measurements}\\&\boldsymbol{\omega}_{\text{correction}}(t) = \text{gyro correction}\end{aligned} \qquad \text{Eqn. 16}$$

Later on we will explain the details of computing the gyro correction vector. Basically, the GPS and accelerometer reference vectors that we have are used to compute a rotational error, which is fed into the computation through the feedback controller, and back into the rotation update equation via equation Eqn. 16.

When we repeat equation 14 for each of the earth axes, we can put the result into a convenient matrix form:

$$\begin{aligned}&\mathbf{R}(t+dt) = \mathbf{R}(t)\begin{bmatrix}1 & -d\theta_z & d\theta_y\\ d\theta_z & 1 & -d\theta_x\\ -d\theta_y & d\theta_x & 1\end{bmatrix}\\&d\theta_x = \omega_x dt\\&d\theta_y = \omega_y dt\\&d\theta_z = \omega_z dt\end{aligned} \qquad \text{Eqn. 17}$$

Equation 17 is a recipe for updating the direction cosine matrix from gyro signals. It is equivalent to Manhoney's result. The values of 1 on the diagonal of the matrix in equation 17 represent the first term in equation 15. The smaller, off-diagonal elements represent the second term in equation 15. Equation 17 is implemented numerically by repeated matrix multiplications, with short time steps. Each matrix multiplication requires 27 multiplications and 18 additions. It maps well to the dsPIC30F4011, which has hardware resources to perform matrix multiplication efficiently. It can be performed on CPUs that do not have matrix support, in which case it is recommended to use integer arithmetic.

The only approximation that equation 17 makes is that the time step is short enough so that the R matrix does not change much from step to step. A typical time step is around 0.020 seconds, during which an aircraft rotating at around 60 degrees per second rotates approximately 0.020 radians, which translates to a maximum change in any of the R matrix coefficients of around 2%. Thus, the second order terms that are being ignored are on the order of 0.02%.

Tests and simulations have shown that implementation of equation 15 by itself, with gyros with modest performance, yields very accurate results that achieve very low drift, on the order of a few degrees per minute. The drift is so low that it is a simple matter to adjust for it without compromising performance. However, by itself, equation 15 will eventually accumulate

numerical round-off and gyro drift, offset, and gain errors. In the next two sections we will explain how to cancel the errors.

Renormalization

Numerical errors will gradually reduce the orthogonality conditions expressed by equation 5 to approximations rather than identities. In effect, the axes in the two frames of reference no longer describe a rigid body. Fortunately, numerical error accumulates very slowly, so it is a simple matter to stay ahead of it.

We call the process of enforcing the orthogonality conditions "renormalization". We devised several ways that it could be done. Simulations showed they all worked quite well, so in the end, we settled on the simplest approach. It works as follows.

First we compute the dot product of the X and Y rows of the matrix, which is supposed to be zero, so the result is a measure of how much the X and Y rows are rotating toward each other:

$$\mathbf{X} = \begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix}$$

$$error = \mathbf{X} \cdot \mathbf{Y} = \mathbf{X}^{\mathrm{T}}\mathbf{Y} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \end{bmatrix} \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix}$$

Eqn. 18

We apportion half of the error each to the X and Y rows, and approximately rotate the X and Y rows in the opposite direction by cross coupling:

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix}_{orthogonal} = \mathbf{X}_{orthogonal} = \mathbf{X} - \frac{error}{2}\mathbf{Y}$$

$$\begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix}_{orthogonal} = \mathbf{Y}_{orthogonal} = \mathbf{Y} - \frac{error}{2}\mathbf{X}$$

Eqn. 19

You can verify that the orthogonality error is greatly reduced by substituting equation 19 into 18, keeping in mind that the magnitude of each row and column of the R matrix is approximately equal to one. Apportioning the error equally to each vector yields a lower residual error after the correction than if the error were assigned entirely to one of the vectors.

The next step is to adjust the Z row of the matrix to be orthogonal to the X and Y row. The way we do that is to simply set the Z row to be the cross product of the X and Y rows:

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix}_{orthogonal} = \mathbf{Z}_{orthogonal} = \mathbf{X}_{orthogonal} \times \mathbf{Y}_{orthogonal} \qquad \text{Eqn. 20}$$

The last step in the renormalization process is to scale the rows of the R matrix to assure that each has a magnitude equal to one. One way we could do that is to divide each element of each row by the square root of the sums of the squares of the elements in that row. However, there is an easier way to do that, by recognizing that the magnitudes will never be much different than one, so we can use a Taylor's expansion. The resulting magnitude adjustment equations for the row vectors are:

$$\begin{aligned} \mathbf{X}_{normalized} &= \frac{1}{2}\left(3 - \mathbf{X}_{orthogonal} \cdot \mathbf{X}_{orthogonal}\right)\mathbf{X}_{orthogonal} \\ \mathbf{Y}_{normalized} &= \frac{1}{2}\left(3 - \mathbf{Y}_{orthogonal} \cdot \mathbf{Y}_{orthogonal}\right)\mathbf{Y}_{orthogonal} \\ \mathbf{Z}_{normalized} &= \frac{1}{2}\left(3 - \mathbf{Z}_{orthogonal} \cdot \mathbf{Z}_{orthogonal}\right)\mathbf{Z}_{orthogonal} \end{aligned} \qquad \text{Eqn. 21}$$

What equation 21 says to do to adjust the magnitude of each row vector to one, is to subtract the dot product of the vector with itself (the square of the magnitude), subtract from three, multiply by ½, and multiply each element of the vector by the result.

There are not that many multiplies and additions in the normalization process. There are no divisions or square roots. We perform the calculation for each step of the integration, every 0.020 seconds.

Drift cancellation

Although the gyros perform rather well, with an uncorrected offset on the order of a few degrees per second, eventually we have to do something about their drift. What is done is to use other orientation references to detect the gyro offsets and provide a negative feedback loop back to the gyros to compensate for the errors in a classical detection and feedback loop, as shown if Figure 1. The steps are:

1. Use orientation reference vectors to detect orientation error by computing a rotation vector that will bring the measured and computed values of reference vectors into alignment.
2. Feed the rotation error vector back through a proportional plus integral (PI) feedback controller to produce a rotation rate adjustment for the gyros. (A PI regulator is a special case of a commonly used feedback regulator called a PID regulator. The D stands for derivative. In our case, we do not need the derivative term.)
3. Add (or subtract, depending on your sign convention for the rotation error) the output of the PI controller to the actual gyro signals.

The main requirement for an orientation reference vector is that it does not drift. Its transient performance is not that important because it is the gyros that provide the transient fidelity for the orientation estimate.

Our two reference vectors are supplied by GPS and accelerometers. Magnetometers are also useful, particularly for yaw control of hovering applications, but for aircraft that fly generally in the direction that they are pointed, a GPS will do just fine. If you use a magnetometer, to provide a vector reference you should use a three axis magnetometer. Low cost three axis magnetometers are commercially available.

We use accelerometers to provide a reference vector for the Z axis of the airplane. Details will be given in a separate section. We use the GPS as a reference for the horizontal projection of the X axis (roll axis) of the plane. Our two reference vectors happen to be perpendicular to each other. That is convenient, but not absolutely necessary.

For either of the two reference vectors, the orientation error is detected by taking the cross product of the measured vector with the vector that is estimated by the direction cosine matrix. The cross product is particularly appropriate for two reasons. Its magnitude is proportional to the sine of the angle between the two vectors, and its direction is perpendicular to both of them. So it represents an axis of rotation, and an amount of rotation, that would be needed to rotate the measured vector to become parallel to the estimated vector. In other words, it is equal to the negative of the orientation rotational error. By feeding it back to the gyros through the PI controller, the estimated orientation is gradually forced to track the reference vectors, and gyro drift is cancelled.

The cross product of a measured reference vector with a corresponding vector computed from the direction cosine matrix is an indication of the error. It is approximately equal to the rotation that would have to be applied to the reference vector to bring it into alignment with the computed vector. We are interested in the amount of rotation correction that we need to apply to the direction cosine matrix, which is equal to the negative of the error rotation. So, it is convenient to compute the correction by interchanging the order in the cross product. The correctional rotation is equal to the cross product of the vector estimated by the direction cosines with the reference vector.

We use a proportional plus integral feedback controller to apply the rotation correction to the gyros, because it is stable and because the integral term completely cancels gyro offset, including thermal drift, with zero residual orientation error.

The way that the reference vector errors map back onto the gyros is done via the direction cosine matrix, so that the mapping depends on the orientation of the IMU. For example, the GPS reference vector might correct either the X, Y, Z, or combinations of X, Y and Z axis gyro signals, depending on the orientation of the axes with respect to the earth frame.

We will now get into more detail for the two references that we are using.

GPS

GPS provides a drift-free reference vector for the yaw orientation of the plane. The only reason that we do not use GPS by itself for yaw information is that the transient response of the gyros is much faster than that of the GPS. Instead, we use GPS as a reference vector to cancel gyro drift and achieve a yaw “lock”.

Two of the major services provided by a GPS radio are reporting of location and velocity magnitude and direction. The GPS determines its location and velocity from the signals that it receives from orbiting satellites, and sends the information out through its serial interface. For most GPS receivers, there are two data formats, NMEA and binary. NMEA is a comma delimited, human readable standardized ASCII format. In the binary interface, binary values are transmitted as sequences of the ones and zeros of their internal binary representation. The binary interface provides some additional information that is not available in the NMEA interface.

The GPS must move in order to give direction information. Otherwise, there is no way to determine the orientation of the GPS antenna. The velocity vector reported by GPS is the change in position of the antenna in 1 second. There are several ways a GPS might do the computation, but for all methods, the GPS must move.

There are two different coordinate systems for GPS units to report location and velocity. One system reports longitude, latitude, altitude, velocity over ground, and course over ground. Course over ground is the angle of the course measured clockwise from the north. Interestingly enough, this is the same angle as measured in the mathematical sense (counter clockwise) around the Z axis in the body reference frame of the plane, with the Z axis pointing down. In this system, vertical velocity is available through the binary interface.

The other system, ECEF (earth-centered, earth-fixed), reports X, Y, Z position and velocity, with the origin of the right-handed X, Y, Z coordinate system at the center of the earth.

A GPS delivers its information as a continuous sequence of reports, typically once every second (1 Hz), though there is a trend toward higher reporting rates, with 5 times a second (5 Hz) becoming more common. However, a higher reporting rate does not necessarily lead to better performance, because of the limitations imposed by the dynamics of the GPS internal signal processing.

There are several factors that should be kept in mind in considering GPS dynamics:

1. Reporting latency. Under certain circumstances, for some GPS units it may take as long as 12 seconds for the computed data to be transmitted.
2. Filtering. All GPS units perform some sort of filtering to improve the accuracy of position and velocity estimation. This will result in a smoothing effect on the data when the GPS changes its velocity or position, so that the new information is not seen instantly, but rather becomes apparent gradually.
3. Track smoothing and static navigation. Many types of GPS radios provide a “track smoothing” option to ignore sudden changes in position or velocity. This is useful for automotive applications to prevent changes from being seen as the result in changes in the satellite signals, such as when collection of satellites that are being used changes. They also provide a “static navigation” option so that variation in the apparent location is suppressed when the velocity falls below a certain value. This is also useful for automotive applications.

It is not likely that you will every run into track smoothing or static navigation, because the factory defaults are to turn these options off, but you should be aware of them. However, reporting latency and filtering must be taken into account.

By reporting latency we mean a simple time delay between when the GPS measures position and velocity, and when it appears in the sequence of messages. Usually this delay is the reporting time period. For example, if your GPS is reporting at 5 Hz, the reporting latency is typically 0.2 seconds. However, it could be much larger than that if you are not careful. One of us (Bill) had the bad luck of stumbling into a 12 second latency with a reporting rate of 1 Hz. It turned out that the 12 second delay was triggered by using a combination of 4800 baud and the binary interface. It was reduced to a 1 second latency by changing the baud rate to 19,200. Chances are that you will not run into this effect, but be aware that it exists. If you use the binary interface, you should use a baud rate of 19,200 or greater.

In addition to a simple latency, you will generally also run into a delay caused by internal filtering done by the GPS. All GPS units perform some sort of filtering of the data by the very nature of how they do their computations. There is an inherent compromise in any system between accuracy and transient response. The more accurate you want to know something, the longer it will take to estimate it. In most units, the filtering shows up as a smoothing of the data. Typically, the dynamic response of many types of GPS is a simple exponential response with a 1 second time constant, so that it takes about 3 seconds to fully respond to a step change. If you ignore the GPS dynamics, there will be a small error introduced into your navigation calculations during a turn. One of us (Paul) saw that it is possible to compensate for this small error by introducing a filter between the direction cosine matrix and the input to the yaw drift correction. [Do we need a figure?].

That way, the dynamics of the two vectors used in the estimation of the yaw error are matched.

It is often assumed that a GPS with a high reporting rate, such as 5 Hz, will provide better dynamic performance that on with the most common reporting rate, 1 Hz. However it is not necessarily the case that the higher reporting rate will provide better dynamic response. Certainly, its latency will be less. However, there is still the issue of the filter dynamics, which will generally turn out to be the limiting effect.

The GPS horizontal course over ground signal has zero drift over the long term, and can be used as a reference vector to achieve “yaw lock” for the IMU. We considered also including the vertical velocity from the GPS, but decided against it, in favor using the accelerometers for vertical information.

The assumption is made that the aircraft is moving in the direction that it is pointing. Any transient errors in that assumption do not materially affect performance. However, strong winds, particularly cross winds, do violate this assumption. There are two approaches that you can take. One approach is to somehow compute the wind vector from available information. We are continuing to work on that. The other approach is to use moderate feedback gains. The difference between the direction the aircraft is pointing and the direction that it is moving will show up as an error at the input to the drift correction feedback controller. The result will be that DCM will adapt to the wind, and rotate the plane the amount required to keep it moving along the desired course over ground.

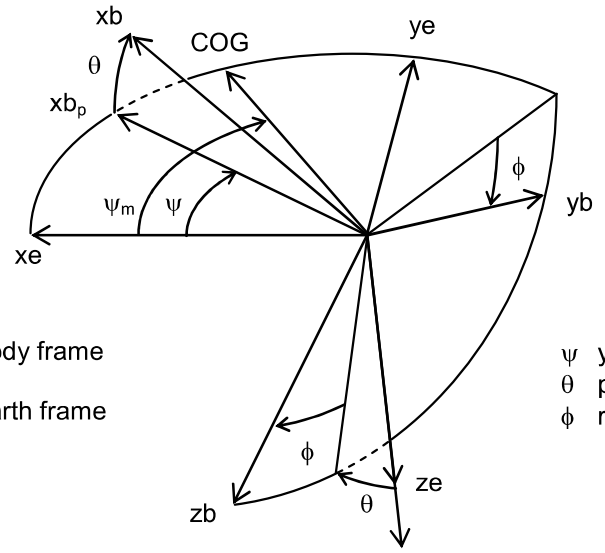
The following figure shows how the yaw correction is computed:

xb_p: projection of xb on earth xy plane

COG: GPS course over ground vector

ψ estimated yaw

ψ_m measured yaw



(xb yb zb) body frame

(xe ye ze) earth frame

ψ yaw angle
θ pitch angle
ϕ roll angle

$\| xb \| = 1 \quad \| xb_p \| = \cos(\theta)$

$xb_p \wedge COG = YawCorrectionGround$

The rotational error between the GPS course over ground vector, and the projection on the horizontal plane of the roll axis (X) of the IMU is an indication of the amount of drift. The rotational correction is the Z component of the cross product of the X column of the R matrix and the course over ground vector.

First, we form the reference vector from the normalized horizontal velocity vector. This can be done by simply taking the cosine and sine of the course over ground angle, in the earth frame of reference:

$$\begin{aligned} COGX &= \cos(cog) \\ COGY &= \sin(cog) \end{aligned} \qquad \text{Eqn. 22}$$

We then compute yaw correction:

$$YawCorrectionGround = r_{xx}COGY - r_{yx}COGX \qquad \text{Eqn. 23}$$

However, equation 23 yields the yaw correction in the earth frame of reference. In order to adjust the gyro drift, we will need to know the correction vector in the aircraft (body) frame of reference. To compute that we must multiply the yaw correction in the ground frame of reference by the Z row of the R matrix:

$$\mathbf{YawCorrectionPlane} = YawCorrectionGround \begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{bmatrix} \qquad \text{Eqn. 24}$$

The yaw correction vector produced by equation 24 will be combined with roll-pitch correction computed from the accelerometers into a total vector that is used to compensate for drift. Details of that computation will be given after we discuss how the accelerometers are used.

There are three conditions relative to yaw drift compensation that argue for a large weighting of the yaw correction, to enable a rapid response to yaw error.

The first condition is initial yaw lock. When the algorithm starts up, it has no way of knowing what direction the board is pointing. Even if it did, during the time it waits for GPS lock, it will be drifting, and even after GPS is locked, the GPS reported course over ground will be random numbers before the plane is launched. By giving the yaw drift correction a large weight, yaw lock can be achieved shortly after takeoff.

The second condition is winds. If the plane travels for a long time in a cross wind, the wind will be treated as a gyro offset. If the plane then makes a 180 degree turn, for a while the DCM algorithm will turn the plane by the opposite angle that it would need to compensate for the wind.

The third condition is when the plane is traveling vertically. During that time the X axis of the plane is vertical, and equation 23 yields zero.

For these reasons, it is best to use a large weight for the yaw drift correction.

Accelerometers

Accelerometers are used for roll-pitch drift correction because they have zero drift. We do have to worry about centrifugal acceleration, but that can be accounted for, and will be discussed shortly.

When one of us (Bill) built his first board, he had hopes that accelerometers could be used by themselves for roll-pitch control. But they cannot, for a number of reasons. The main reason is that they measure a combination of acceleration and gravity. If they measured only gravity, they would be perfect. But they measure acceleration, too, and that can cause trouble. Bill once tried to use accelerometer-only based pitch stabilization during a hand launch of a sailplane. The acceleration of the launch fooled the controls into estimating that the plane was pitching up. The controls responded by pitching the plane straight down.

The way an accelerometer typically works is that it measures the deflection of a small mass suspended by springs. The natural frequencies of the dynamics of the accelerometer are high, so it does respond quickly. The deflection depends on the total force on the mass, which is equal to its mass times the sum of the gravity vector plus the acceleration vector. (The usual sign convention for accelerometers is such that they indicate gravity minus the acceleration.)

So in addition to gravity, an accelerometer also measures acceleration. That should not be too surprising, since that is what they are called. Therefore, an accelerometer is useful as a roll-pitch indicator only when the plane is not accelerating. The problem is it is often accelerating. Some of the accelerations, such as centrifugal acceleration, are easy enough to compute and compensate for without having a model of the dynamics of the plane. However, there is no easy way to separately compute the forward acceleration.

All is not lost. On average, a plane does not accelerate in the forward direction. There are times when it speeds up and when it slows down, but the accelerations cancel out. A plane cannot accelerate for long in the forward direction until aerodynamic drag prevents it from going any faster. A plane cannot decelerate for long without stopping and falling from the sky. As long as we are not depending on an accelerometer for fast transient response, we can use it for roll-pitch correction of gyro drift, because the accelerometer does not drift.

There are many good accelerometers on the market, most of them will work just fine with the DCM algorithm. They are not as critical as gyros, because any change in their offsets does generate an accumulated error in the way that a gyro offset does. An accelerometer is a direct measurement of orientation, while a gyro is a measurement of the time rate of change of orientation.

There are a variety of interface types, including analog voltage, pulse width modulation, and several standard communications interfaces. We chose an accelerometer with an analog voltage output as the simplest interface.

The greatest advantage of using direction cosines is that they work for any orientation of the plane, without any singularities or special logic. Any orientation can be well-described by the 9 elements of the direction cosine matrix. Since we will need to perform the drift cancellation calculations for any orientation of the plane, we will need to measure acceleration along all three axes of the plane. This can be done with commercially available 3 axis accelerometers, or with 3 separate units.

Before we can use the accelerometer information for roll-pitch drift compensation, we must account for the centrifugal acceleration associated with changes in direction of the planes forward velocity. Although a plane can accelerate or decelerate along the forward direction for a short while, it can turn indefinitely.

Fortunately, the information needed to compute the centrifugal acceleration is readily available. Centrifugal acceleration is equal to the cross product of the rotation rate vector with the velocity vector. We do not need an exact answer, only one that is accurate on average. On average, the plane moves in the direction that it is pointed. Therefore, we can assume that the velocity vector is parallel to the X axis of the plane. GPS gives us the magnitude of the velocity over ground. Since ground is an inertial reference frame, we can

compute the velocity vector in the plane (body) frame of reference as being the velocity over ground, in the X direction.

In the plane (body) frame of reference, we compute the centrifugal acceleration as the cross product of the gyro vector and the velocity vector:

$$\mathbf{A}_{\mathbf{centrifugal}} = \boldsymbol{\omega}_{\mathbf{gyro}} \times \mathbf{V}$$
$$\mathbf{V} = \begin{bmatrix} velocity \\ 0 \\ 0 \end{bmatrix} \qquad \text{Eqn. 25}$$

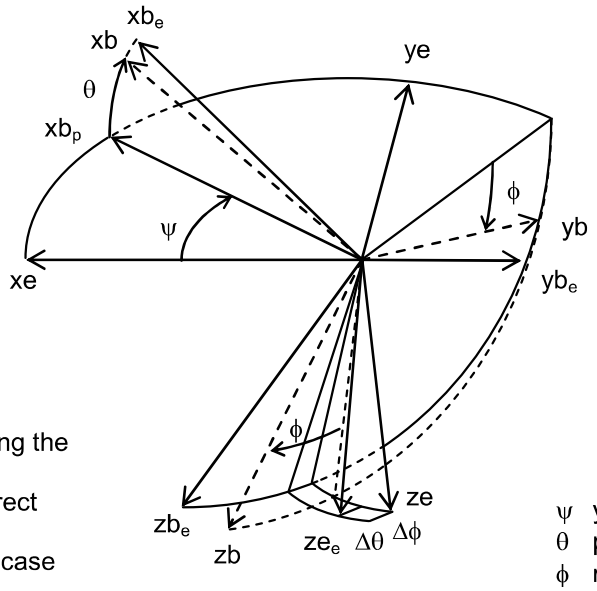
Note that in equation 25, we only need to perform two multiplications, because two of the elements of the velocity vector in the plane (body) frame of reference are zero.

The usual sign convention for commercial three axis accelerometers is that the Z axis points down, and the downward pull of gravity produces a positive output. Therefore, the output of the accelerometers is gravity minus the acceleration. To recover an estimate of gravity that is adjusted for centrifugal acceleration, we need to add the centrifugal acceleration estimate. Therefore, the reference measurement of gravity in the plane (body) frame is given by:

$$\mathbf{g}_{\mathbf{reference}} = \mathbf{Accelerometer} + \boldsymbol{\omega}_{\mathbf{gyro}} \times \mathbf{V}$$
$$\mathbf{Accelerometer} = \begin{bmatrix} Accelerometer_x \\ Accelerometer_y \\ Accelerometer_z \end{bmatrix} \qquad \text{Eqn. 26}$$

In addition to the reference measurement of gravity, we need an estimate based on the direction cosine matrix. It is furnished by the Z row of the direction cosine matrix, which is the projection of the earth frame of reference "down" axis along the axes of the plane (body) frame of reference.

(xb_e yb_e zb_e)	estimated body frame	ze_e	estimated earth z vector
(xb yb zb)	body frame	ze	earth z vector



On this drawing the yaw angle is assumed correct but this is not generally the case

ψ yaw angle
θ pitch angle
ϕ roll angle

The roll-pitch rotational correction vector in the body frame of reference is computed by taking the cross product of the Z row of the direction cosine matrix with the normalized gravity reference vector:

$$\mathbf{RollPitchCorrectionPlane} = \begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{bmatrix} \times \mathbf{g}_{\mathbf{reference}} \qquad \text{Eqn. 27}$$

During very tight, continuous turns, the accelerometers might become saturated. In other words, the actual acceleration might exceed the range of the accelerometer. In that case, error will be introduced into the roll-pitch orientation estimate. The controls should be designed to avoid saturating the accelerometers. Similarly, the gyros can become saturated during rapid turns. That can be avoided by including gyro terms in the control feedback to limit the turning rate.

Feedback controller

Each of the rotational drift correction vectors (yaw and roll-pitch) are multiplied by weights and fed to a proportional plus integral (PI) feedback controller to be added to the gyro vector to produce a corrected gyro vector that is used as the input to equation 17. (Now is a good time to go back an look at Figure 1.) The calculation proceeds as follows. First we compute a weighted average of the total of the rotation corrections. In our case, there are just two corrections, but in general there could be more:

$$\mathbf{TotalCorrection} = W_{RP}\mathbf{RollPitchCorrectionPlane} + W_Y\mathbf{YawCorrectionPlane} \quad \text{Eqn. 28}$$

Next, we pass the total correction through a PI controller:

$$\begin{aligned} \boldsymbol{\omega}_{\mathbf{PCorrection}} &= K_P\mathbf{TotalCorrection} \\ \boldsymbol{\omega}_{\mathbf{ICorrection}} &= \boldsymbol{\omega}_{\mathbf{ICorrection}} + K_I dt\mathbf{TotalCorrection} \\ \boldsymbol{\omega}_{\mathbf{correction}} &= \boldsymbol{\omega}_{\mathbf{PCorrection}} + \boldsymbol{\omega}_{\mathbf{ICorrection}} \end{aligned} \quad \text{Eqn. 29}$$

We then feed the gyro correction vector back into the rotation update equation by adding the correction vector to the gyro signal, as shown in Eqn. 16.

At this point, be have completed a full pass through the calculation. At the next time step we repeat the entire calculaton.

Some readers may be wondering why we use a single feedback controller with weighted inputs rather than separate controllers for each of the two vectors. Actually, we could, except over a long period of time the separate integrators could accumulate equal and opposite errors that could eventually cause the integrators to saturate or roll over. Tests have shown that that would take a very long time. However, it is more correct to use a single controller.

Selection of the weights and gains is a compromise between accuracy and speed of recovery to disturbances. The practical realities of the wind and gyro saturation favor using weights and gains that are large enough to recover in about 10 seconds. In the feedback loop, the DCM algorithm is a nonlinear integrator. Therefore, you can select the gains for the linearized equivalent dynamic model of the complete loop.

Gyro characteristics

Gyro sensitivity, operating range, offset, drift, calibration, and saturation must be taken into account in the implementation of DCM.

- Gyro sensitivity – Usually expressed in millivolts per degree per second for a gyro with an analog output, gyro sensitivity is the gyro gain for converting rotation rate to a voltage. In the early phases of the development of DCM, it was thought that sensitive gyros, on the order of 15 millivolts per degree per second, were needed because they usually had low offset and drift. It turns out that DCM works well with other units with lower sensitivity. Gyro sensitivity is related to operating range. The more sensitive the gyro, the narrower the useful operating range and vice-versa. Gyro sensitivity must be taken into account for gyro calibration. Some analog gyros provide an output voltage that is referred to an absolute voltage reference. If such gyros are measured with a ratiometric A/D, then you should measure a known voltage reference to account for an apparent

dependency of the sensitivity and offset of the gyro with supply voltage. Other analog gyros may provide a ratiometric referenced output, in which case you use a ratiometric A/D, then you do not need to adjust for supply voltage variation.

- Offset – The gyro offset is its output when there is no rotation. Gyro offset varies from unit to unit and also may vary with temperature and supply voltage. Most of the offset can be removed by simply measuring the offset during power up, provided you keep the gyros motionless at that time. The variation of offset with supply voltage and temperature is usually rather slow, so that DCM can continually remove the offset and maintain lock.
- Drift – By drift we mean the integrated effects over time of a slowly varying offset and noise. Drift around all three axes can be completely eliminated with DCM as long as the aircraft continues to move forward. If it stops moving, there will be a yaw drift that depends on the residual offsets of the gyros. When it starts moving, the drift will be cancelled again in a few seconds.
- Calibration – By calibration we mean applying the correct gain multipliers to the gyro signals before applying the update algorithm. Since the update algorithm is basically a nonlinear integrator, if the gains are set too high, the DCM computations will "over-rotate" during a continuous turn. If they are too low, DCM will "under-rotate". We found that you can operate DCM without setting the gains exactly. They do have to be set approximately correct in order to achieve drift lock. Nominal calibration works well enough as long as the feedback gains are large enough, in which the error that begins to accumulate during a continuous turn is treated as an offset and it is compensated for.
- Saturation – If the rotation rate of the aircraft exceeds the maximum range of a gyro, the gyro will "saturate". This will generate an angular estimation error equal to the area between the actual rate and the saturated rate. There several practical ways of addressing this issue. The simplest solution is to use feedback gains that are large enough to erase the error in a reasonable amount of time, on the order of 10 seconds, for example. This will still retain the “smoothness” and overall accuracy of IMU control. A second solution is to include some gyro feedback in the controls to reduce the rotation rate for the axis most likely to saturate. A third solution is to implement the full version of Mahony’s approach, and to integrate estimation with control, with a constraint applied to the desired turning rates.

Wind

The effects of wind must be considered, mostly with respect to yaw. Since we are using course over ground to achieve yaw lock, the direction cosine matrix axes will eventually align with the direction the plane is headed, not the

direction that it is pointed. This is fine when the plane is headed in a straight course back to RTL, or between way points if you use DCM for an autopilot. What will happen in that case is that the algorithm will treat the wind as drift, and gradually rotate the plane by the angle that is required to maintain the desired course over ground. However, when the plane makes a turn, it will take a finite amount of time for DCM to adapt to the new angle between the wind direction and the course over ground. There are two solutions to the problem:

- Use feedback gains that are large enough to adapt to the wind within a few seconds after a change in wind or in course. This is the approach that we are presently using in our firmware.
- Somehow compute the wind vector. In principle this should be possible to do, given the low residual gyro offset, provided the plane makes some turns. We are presently looking into this approach, to see if it will work better than the above approach. For now, we will continue to adapt to the wind after a turn, because that is actually working rather well.

Using DCM in control and navigation

In a previous section we described several applications of direction cosines for control and navigation. In this section we provide some more detail:

1. To control the pitch of an aircraft, you need to know the pitch attitude of the aircraft, which you can find by taking the dot product of the roll axis of the aircraft with the ground vertical.

The dot product of the roll axis (X) of the aircraft with the ground vertical (Z) is one of the direction cosines, r_{zx}. It is equal to the sine of the angle between the roll axis and the horizontal plane in the earth frame of reference. So that element of the matrix is a direct indication of whether or not the roll axis of the plane is parallel to the ground, and can be used directly in a feedback loop to control pitch. When the plane is level, r_{zx} will be equal to zero.

2. To control the roll of an aircraft, you need to know the bank attitude of the aircraft, which you can find by taking the dot product of the pitch axis of the aircraft with the ground vertical.

The dot product of the pitch axis (Y) of the aircraft with the ground vertical (Z) is one of the direction cosines, r_{zy}. It is equal to the sine of the angle between the pitch axis and the horizontal plane in the earth frame of reference. So that element of the matrix is a direct indication of whether or not the pitch axis of the plane is parallel to the ground, and can be used directly in a feedback loop to control pitch. When the plane is level, r_{zy} will be equal to zero.

3. To navigate, you need to know the yaw attitude of the aircraft with respect to the direction that you want to go, which you can find by

taking the cross product of the roll axis of the aircraft with a vector in the direction that you want to go. This works even if you are upside down. To find out if your aircraft might be pointing in the opposite direction than you want to go, take the dot product of the roll axis with the desired direction vector. If it is negative, the aircraft is more than 90 degrees off course.

The roll axis vector of the aircraft is the first column of the R matrix. We only want the horizontal components, so we set the Z component to zero. The resulting vector is equal to $\begin{bmatrix} r_{xx} & r_{yx} & 0 \end{bmatrix}$. We take the cross product of that vector with the desired direction vector to get the sine of the deviation angle, and we take the dot product to get the cosine.

4. To find out if the aircraft is upside down, examine the sign of the dot product of the aircraft yaw axis with the vertical. If it is less than zero, the aircraft is upside down.

The dot product of the aircraft yaw axis with the vertical is the matrix element r_{zz}. When the plane is flying more or less level, this element is approximately one. When the plane is upside down, this element is approximately minus one. When the plane is banked sideways at a 90 degree angle, this element is zero.

5. To find out the turning rate of the aircraft around the vertical earth axis, transform the gyro rotation vector to the earth frame of reference, and take the dot product with the vertical axis.

This is equivalent to taking the dot product of the third row of the R matrix with the gyro rotation vector. So, the turning rate of the aircraft around the vertical earth axis is equal to $\omega_x r_{zx} + \omega_y r_{zy} + \omega_z r_{zz}$.

Implementation

We are planning to write a separate paper on how to implement the DCM algorithm in C code. Some readers may have access to firmware that we have written. If so, to avoid confusion, you should be aware of the following:

This paper expresses all quantities using an aviation convention for the 3 axes. X is forward, Y points along the right wing, and Z is down. However, the firmware that we have written uses a coordinate system that is somewhat different. X points along the left wing, Y is forward, and Z is down. The reason that we did this is because in the design of the board, the three axis accelerometer was mounted in such a way that when you mount the board in a plane with the longer dimension aligned in the most convenient orientation, the accelerometer, and the labels on the board, point the Y axis forward.

References

Several papers written by Robert Mahony et al are available at http://gentlenav.googlecode.com/files/MahonyPapers.zip.