A Dynamic stability analysis rocket simulator

Introduction:

The basic static stability of the fixed-fin rocket vehicles that we fly has been covered by Barrowman's superb analysis (Ref's 4, 5), and others, and won't be repeated here. See our paper 'Rocketry aerodynamics' for details.
(Note that several books and web-pages contain errors in their reprinting of the Barrowman stability equations; it's better to get a copy of the original paper, such as from the Apogee rockets website.)

The Barrowman method is a classic static-stability analysis: it simply tells you whether your fins are large enough so that your vehicle has a tendency to keep the nose pointing in the direction of flight as required, and it assumes that the ensuing rotation of the vehicle about its Centre of Gravity (CG) is slow enough not to affect the analysis.

Other questions can only be answered by extending the static stability analysis to include the possibility of rapid rotation of the vehicle about its CG; this is known as a dynamic stability analysis.
These questions might be: how long will the vehicle take to bring the nose back to true after being hit by a gust? How big an angle of attack will it pull while doing this? Will the nose wag from side to side a lot during the flight causing extra drag?

Also, and most importantly, when the rocket vehicle leaves the launcher and encounters the wind, how much will the wind bend the trajectory?

Shameless plug: In my book 'Rocket science and spaceflight for young rocketeers' I showed how to write a simple rocket simulator on an Excel spreadsheet, but it was limited to two degrees of freedom (a non-rotating point mass).
In this paper, I'll describe a 3 degree-of-freedom (includes pitch rotation) rocket simulator that you can write to answer the above questions.

Nomenclature:

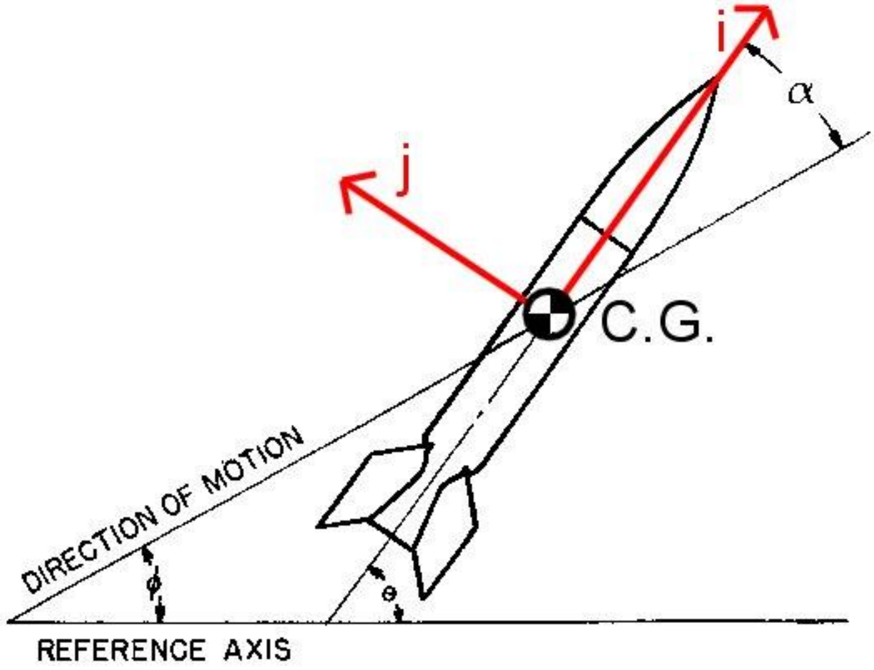
θ = Pitch angle (radians)

$\dot{\theta}$ = Pitch rate (radians/second). The dot is Newton's notation for time rate of change.

ϕ = Climb angle (also known as flight path angle or trajectory angle), (radians)

α = Angle of attack (radians)
ρ = Atmospheric density (kg/m^3)
A = Acceleration (m/sec^2)
CH = damping moment
CM = Moment coefficient (dimensionless)
CN = Normal force coefficient (dimensionless)
CN_α = Gradient of Normal force coefficient per radian angle of attack (1/radian)
d = Diameter of (thickest part of) the fuselage (m)
I = moment of inertia (kgm^2)
m = mass (kg)
M = moment (Nm)
N = normal force (N)
S = Cross-sectional area of (thickest part of) the fuselage (m^2)
V = Velocity (m/sec)

Axes system:
Before we start we must define a system of axes. For rocketry, we usually use right-handed i,j,k, body axes, where the i-axis is the axis of symmetry of the vehicle:



Positive pitch angle θ is nose-up, positive climb angle ϕ (flight path angle) is above the horizon. α is the angle of attack, and is traditionally always positive.

Positive pitch rate $\dot{\theta}$ is nose rising. The direction of pitch rate (which we'll need later) is defined to be along the axis of pitch rotation, which is the 'k' axis, which according to the right hand grip rule (Wikipedia) is coming toward you out of the page here.

The wind
The next biggest deflector of the vehicle's trajectory (after gravity) is a side-wind.
In our sim, we'll switch on the wind (blowing from the right in the above picture) at the instant the vehicle leaves its launcher.
For ease of mathematics, we'll split the wind into two components: blowing along the reverse of the i body axis direction, and along the reverse of the j body axis direction.
The airspeeds of the vehicle in (the reverse of) i,j, axes are then:

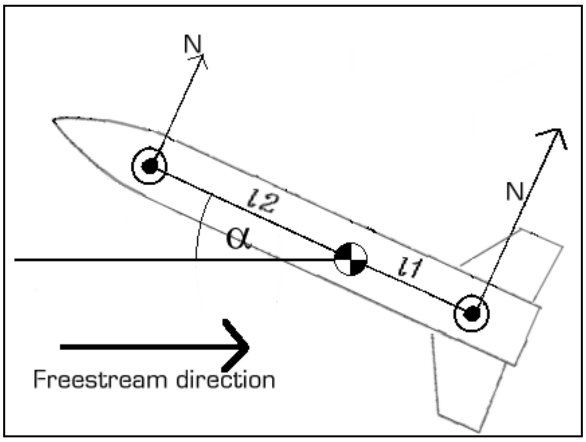
$$Vu = \quad Vi + wind\ from\ the\ right \cos\theta \qquad (1.1)$$
$$Vv = -Vj + wind\ from\ the\ right \sin\theta \qquad (1.2)$$

Where *Vi* and *Vj* are the velocities of the vehicle in the i and j axes directions: We have yet to calculate these, so we'll use the standard sim trick of using the answers from the last time round the simulation loop, from the last time step. (In a sim, we repeat the same equations again and again, increasing time by a small increment for each repeat.)

The vehicle's airspeed is then obtained from Pythagoras: $V = \sqrt{Vu^2 + Vv^2}$ (1.3)

Static stability

A little review of static stability:
The angle of attack, α, is the angle that the incoming air meets the nose and fins. (Barrowman's analysis assumes that this angle is very small, which is a valid assumption during most of the rocket vehicle's flight).
The angle of attack causes Lift (A Normal force 'N') at the nose and tailfins.



Take the distance between the CG and the fins Centre of Lift (Centre of pressure CP_{fins}) as l_1, and the distance between CG and forebody (nose) CP_{fore} as l_2 as shown here:

The total aerodynamic moment *M* about the CG is then:

$N_{fins} l_1 - N_{forebody} l_2$ (1.4) A positive moment is nose-up.

And so an aerodynamic moment coefficient can be defined:

$$CM_{CG} = -\frac{CN_{fins} l_1 - CN_{forebody} l_2}{d} \quad (1.5) \quad \text{(therefore } M_{CG} = \tfrac{1}{2}\rho V^2 S\, d\, CM_{CG}\text{)} \quad (1.6)$$

The minus sign indicates that this 'restoring moment' opposes any increasing angle of attack for a stable rocket.
The '*d'* is a reference length used to make the moment coefficient dimensionless like the other aerodynamic coefficients; the fuselage diameter that was used to calculate the reference cross-sectional area *S* is used as '*d*' for consistency. As you can see, the '*d*' drops out when you multiply equation (1.2) by (1.3), it's superfluous.

Reference 6 calls this moment (equ. 1.3) the Corrective moment coefficient C1 (though it is *not* a coefficient, it actually has dimensions, and Ref. 6 uses the opposite sign for direction of rotation).

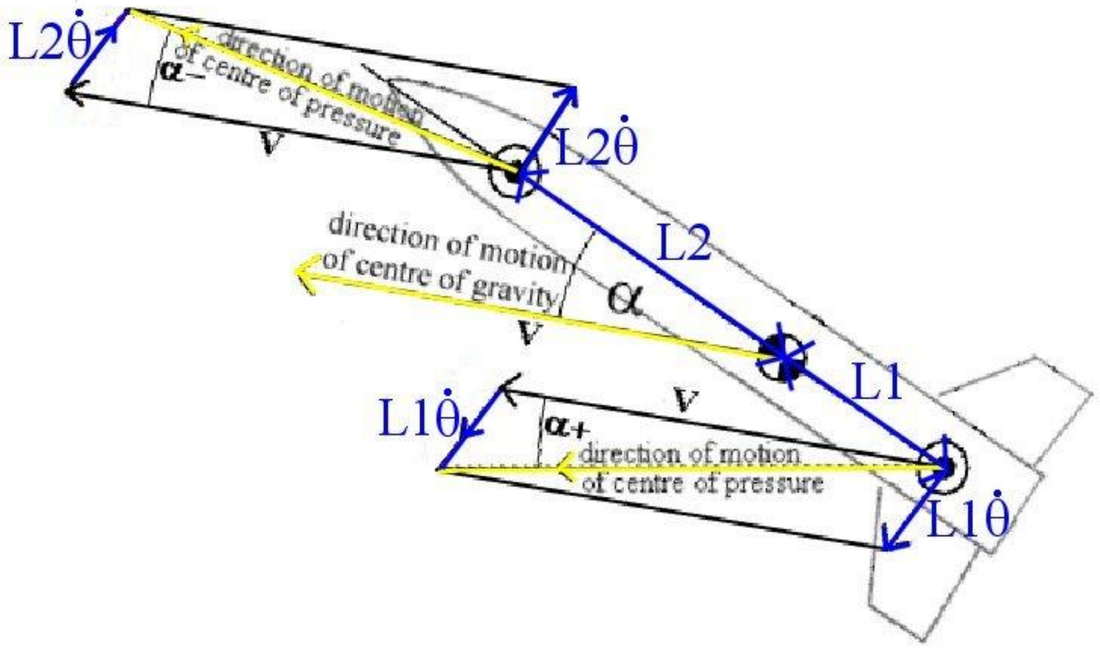
This corrective moment equation describes the 'static' stability, so-called because it is assumed that the vehicle rotates extremely slowly (technically infinitely slowly; rotationally static) about the CG.

A faster rotation rate causes further 'dynamic' effects. The rotation is a pitch rotation, with positive pitch rate giving a nose-up rotation.

Reference 6 is a good attempt to analyse the dynamics of rocket flight, and I believe it forms the basis of dynamic analyses used by the excellent Rocksim software.
However, reference 8 shows that an important feature is missing from that analysis, namely that the rocket vehicle is also able to move laterally as it rotates. The effect of this extra degree of freedom is shown in Ref 8 to increase the natural 'tail wagging' frequency and more importantly to increase the damping factor (see below): Reference 6 incorrectly assumes that vehicles with a high pitch moment of inertia will have near-zero rotational damping.
In our sim, we'll allow the vehicle to move laterally as it rotates.

Dynamic effects

The Angle of attack on the nose, fin, (or boat-tail) is defined to be the angle between the incoming 'wind' and the long body axis (i) of the vehicle.
In the diagram above, the incoming 'wind' had the same direction all along the vehicle, so a general constant fuselage angle of attack was used, α_{CG}
However, if the vehicle is rotating in pitch, the direction of the incoming 'wind' varies along the fuselage due to the rotation, so the angle of attack must vary along the fuselage too.



A pitch-up rotation about the CG causes the fins to revolve with a velocity $l_1\dot{\theta}$ where $\dot{\theta}$ is the pitch rate, and the forebody similarly revolves with a velocity $l_2\dot{\theta}$

Adding the airspeed vector *V* (of the CG), the resulting velocity vector diagram above shows that the directions of motion (in yellow) of the fins and forebody are different to that of the motion direction of the C.G.
Remember that these directions of motion are exactly opposite to the directions of the incoming 'wind' at each point, so the angle of attack of the (centre of pressure of the) forebody is reduced by an amount α- as shown, whereas the fins (and boat-tail) angles of attack are increased by an amount α+

So rotating the vehicle nose-up reduces the lift on the nose, and increases the lift on the tailfins: this is an added stabilising effect that opposes the rotation, and is called Aerodynamic damping.

Aerodynamic damping

Let's investigate this damping mathematically.
Firstly, we'll assume that the extra angles of attack are small, so that we can use small angle approximations :

$$\cos(\alpha) \approx 1, \quad \sin(\alpha) \approx \tan(\alpha) \approx \alpha \qquad \text{(radians)}$$

It can further be assumed that for small angle of attacks the centre of pressures (CPs) of the fins and forebody don't move appreciably as α_{CG} , so that l_1 and l_2 can be assumed fixed.

(They will slowly change with time due to movement of the CG as propellant is expelled out the rocket nozzle; do recalculate them every time round the simulation loop.)

Furthermore, if we assume that α_{CG} is small (which assumes that the vehicle left the launcher with a goodly airspeed before being hit side-on by the wind) then the blue rotation vectors in the above picture become less skewed (more of a right-angle) to the velocity vector V.

With all of these assumptions, the extra 'dynamic' angles of attack are simply

$$\alpha^{+} = \frac{l_1\dot{\theta}}{V} \qquad \alpha^{-} = \frac{l_2\dot{\theta}}{V} \qquad (1.7)$$

where *V* is the vehicle's airspeed. (use l_3 for the boat-tail)

Note that at the moment of liftoff, or at apogee on an exactly vertical flight, then *V* will be either zero or very small. This will cause either a divide-by-zero error on a computer, or if *V* is very small will make the dynamic angles of attack huge. To prevent these, limit *V* to be always greater than or equal to 1.0 in equations 1.7

To work out the main 'static' angle of attack α_{CG} we'll use the two 'incoming air' components that we calculated earlier: along the reverse of the i body axis direction (*Vu*) and along the reverse of the j body axis direction (*Vv*)

So: $\alpha_{CG} = \tan^{-1}\left(\frac{Vv}{Vu}\right) \qquad (1.8)$

The resulting angle of attack on the forebody is then:

$$\alpha_{fore} = \tan^{-1}\left(\frac{Vv}{Vu}\right) - \frac{l_2\dot{\theta}}{V} \qquad (1.9)$$

And on the fins is:

$$\alpha_{fins} = \tan^{-1}\left(\frac{Vv}{Vu}\right) + \frac{l_1\dot{\theta}}{V} \qquad (1.10)$$

And on the boat-tail is:

$$\alpha_{boat} = \tan^{-1}\left(\frac{Vv}{Vu}\right) + \frac{l_3\dot{\theta}}{V} \qquad (1.11)$$

The lift coefficient (strictly called the Normal force coefficient in body axes) of the forebody is simply:

$CN_{forebody} = CN_{\alpha}\alpha_{forebody} \qquad (1.12)$ where CN_{α} is the Normal force coefficient of the forebody per radian, which is output from the Barrowman analysis.

And the Normal force coefficient of the fins is similarly:

$CN_{fins} = CN_{\alpha}\alpha_{fins} \qquad (1.13)$ where CN_{α} is the Normal force coefficient of the fins per radian, output from the Barrowman analysis.

For the boat-tail, the Normal force coefficient of the boat-tail per radian which is output from the Barrowman analysis is actually negative, so the Normal force is negative (a suction).
$CN_{boattail} = CN_{\alpha}\alpha_{boattail} \qquad (1.14)$

The total Normal (lift) force acting on the vehicle is then:

$$N = \frac{1}{2}\rho V^2 S\left(CN_{forebody} + CN_{fins} + CN_{boattail}\right) \qquad (1.15)$$

These lifts (Normal forces) cause a pitching moment (+ve nose-up) about the C.G. which is:

$$M = \frac{1}{2}\rho V^2 S\left(CN_{forebody} l_2 - CN_{fins} l_1 - CN_{boattail} l_3\right) \qquad (1.16)$$

Note that we have a problem with the fins centre of pressure l_1: The fins centre of pressure given by ESDU sheets (see our paper 'Rocketry aerodynamics' depends upon fin angle of attack, but the added dynamic angle of attack on the fins depends upon the fin centre of pressure, it's a circular relationship.
Microsoft Excel will handle circular references, but you have to switch this ability on by going to the 'File' tab (or the 'Tools' option in older versions of Excel) then click 'Options', and then click 'Formulas'.
In the 'Calculation options' section, select the' Enable iterative calculation' check box.

In our trajectory simulator, this circular relationship will be handled by using the value of angle of attack from the last time iteration (last time round the calculation loop) to calculate the current centre of pressure, which then allows the calculation of angle of attack for the current time iteration.

Cross-spin force and damping moment

Recall equations (1.7):

$$\alpha^{+} = \frac{l_1\dot{\theta}}{V} \qquad \alpha^{-} = \frac{l_2\dot{\theta}}{V}$$ where V is the vehicle's airspeed.

The lift that these equations (1.7) angles of attack cause are sometimes called the Cross-spin force coefficients *CS* (Ref.1) although they aren't simple coefficents.
Note that as V is on the denominator of these fractions, then the faster the vehicle is flying, the smaller these extra angles of attack get, so the aerodynamic damping of rockets and aeroplanes decreases at high speed. This is why the pilots of Spaceship One had to be on their toes at high airspeeds of up to Mach 3!
You can formulate similar equations for damping in the roll axis, which again disappears at high airspeeds, which is why high-altitude finned rocket vehicles tend to spin rapidly.

In our pitching moment equation above, the CN's include these 'dynamic' angles of attack. However, the moment caused by these angles of attack alone are often treated separately: In reference 6 for example, equations (1.7) are used to create a separate moment about the centre of gravity of the vehicle called the Damping moment coefficient C2A (Ref.6), or damping moment *CH (*Ref.1*).* Again, it isn't actually a coefficient.

From equ.s (1.7), the damping moment is:

$$CH = \frac{1}{2}\rho V^2 S\left(l_1 CN_{\alpha fins}\frac{l_1\dot{\theta}}{V} + l_2 CN_{\alpha forebody}\frac{l_2\dot{\theta}}{V} + l_2 CN_{\alpha boattail}\frac{l_3\dot{\theta}}{V}\right) \qquad (1.17)$$

$$= \frac{1}{2}\rho V S\dot{\theta}\left(l_1^2 CN_{\alpha fins} + l_2^2 CN_{\alpha forebody} + l_3^2 CN_{\alpha boattail}\right) \qquad (1.18)$$

But this is all rather a digression just to explain where equation (1.18) as used in ref. 6 comes from. In our sim we'll use equations (1.9) to (1.11), and we'll run the sim at at least 1000 Hertz (0.001 second time increments) to capture the pitch oscillations with sufficient resolution.

Other aerodynamic effects

You may be wondering whether my sim could be used by the bad guys. Fortunately no, because there are other, admittedly smaller, aerodynamic effects that our simple sim can't include, simply because we don't have the windtunnel data.

Firstly, the rate of change of angle of attack is actually a complex equation to get correctly, and our vehicles do respond aerodynamically to angle of attack rate.

Secondly, there's the can of worms known as unsteady aerodynamics:
The usual aerodynamic equations assume 'steady flow': 'steady' (or rather 'quasi-steady') means that the shape of the flowfield around the vehicle varies only very slowly with time.
For a vehicle undergoing rapid pitch and oscillations this clearly isn't true, but what constitutes a 'fast' or 'unsteady' flowfield as opposed to a 'quasi-steady' one is ill defined.

The upshot is that the *CN*s, and therefore the *M* may be higher than even the dynamic terms above suggest.
By how much we don't know, so we just have to assume steady for now.
Fortuitously, it would appear that unsteady aerodynamics will increase the aerodynamic damping of pitch rotation.

So our sim, as with all sims, is limited by the available aerodynamic data we can feed-in. Fortunately, sims such as Rocksim show that despite this, the sim results and trends are useful.

Rocket exhaust effect: Jet damping

Another major damping term comes courtesy of the rocket nozzle exhaust:

$$Jet\ damping = -\dot{m}\dot{\theta}\, r_e^2$$ (1.19, see derivation in Appendix 2 at the end of this paper)

Where $\dot{m}$ is the mass flow rate of exhaust gasses out of the nozzle in kg/second and r_e is the distance from the center of gravity of the rocket to the exit of the nozzle (metres).
This is the equation we'll code into our sim and add to the aerodynamic pitching moment from equ. (1.16)
Remember to recalculate r_e every time round the simulation loop as it will slowly change with time due to movement of the CG as propellant is expelled out the rocket nozzle.

If you have a thrust curve but not a nozzle mass flow rate curve, then there's a simple assumption you can make that will give you this curve: assume that the effective exhaust velocity of the rocket is constant with time. Several popular rocketry sims make this assumption.

Now: thrust = nozzle mass flow rate × effective exhaust velocity

So if the effective exhaust velocity is constant, then the shape of the thrust curve will be identical to the shape of the nozzle mass flow rate curve.

Because the two graphs are identical, then the areas under their curves are related:

$$\frac{area\ under\ thrust\ curve}{area\ under\ mass\ flow\ rate\ curve} = \frac{total\ impulse}{total\ propellant\ mass}$$ = *effective exhaust velocity*

So now that we have the effective exhaust velocity, then simply re-arrange the thrust equation to get:

$$nozzle\ mass\ \ flow\ rate = \frac{thrust}{effective\ \ exhaust\ velocity}$$

Equations of motion

Now that we have calculated the forces and moments on the vehicle, it's time to crunch the numbers and calculate how the vehicle moves.

Firstly the resultant force along the 'i' axis of the vehicle:

Axial force = thrust – Drag (2.1) where for small α_{CG}, $Drag = \frac{1}{2}\rho V^2 S C_D$ (2.2)

$$Axial\ acceleration = \frac{axial\ force}{vehicle\ mass} \quad (2.3)$$ and

$$Normal\ acceleration = \frac{Normal\ force}{vehicle\ mass} \quad (2.4)$$

Remembering to recalculate the vehicle mass each time round the sim loop as propellant is expelled out the nozzle (see appendix 1).

Next, we subtract the effect of gravity (g = 9.81 m/sec^2):

$$Axial\ acceleration = equ.(2.3) - g \sin\phi \quad (2.5)$$
$$Normal\ acceleration = equ.(2.4) - g \cos\phi \quad (2.6)$$

The theorem of Coriolis

At this stage, we have the Normal and Axial accelerations, which are in the body axes of the vehicle. However, this axis system is rotating in pitch with respect to the world as the vehicle rotates.
Coriolis's theorem is an equation that compensates for the effect of rotating axes, and adds the extra effects that rotating axes cause. (For example, centrifugal 'force' is caused purely by rotating axes).
One of the best proofs I've come across for the Coriolis theorem is the simple fact that if you ignore the following, your sim will go nuts: the vehicle's trajectory goes all over the place. Whereas if you code the following, all will be well:

The Coriolis theorem is written as a rather mathematically heavy equation, which is why I wasn't taught it 'till University, but fortunately the eventual answer is very simple. So grit your teeth and proceed:

Acceleration is the time differential of velocity, and velocity is the time differential of distance. Coriolis's theorem deals with these time differentials:

$$\frac{d}{dt}[X]_{\substack{non-rotating\\bodyaxes}} = \frac{d}{dt}[X]_{\substack{rotating\\bodyaxes}} + [\omega]\times[X] \quad (2.7)$$

where $[\omega]$ is the angular velocity vector of the rotating axes and $[X]$ is the vector (velocity or displacement) to be adjusted for the effects of this rotation.

The '$\times$' isn't a multiplication sign, instead it signifies a process called the vector product (see http://en.wikipedia.org/wiki/Cross_product)

In our rotating i,j,k axes (where pitch rotation is along the 'k' axis), this equation becomes in matrix form:

$$\begin{bmatrix} Ai \\ Aj \\ 0 \end{bmatrix}_{non-rotating\ axes} = \begin{bmatrix} Ai \\ Aj \\ 0 \end{bmatrix}_{rotating\ axes} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}k \end{bmatrix} \times \begin{bmatrix} Vi \\ Vj \\ 0 \end{bmatrix} \qquad (2.8)$$

Where the *A*'s are the body-axes accelerations, and the *V*'s are the vehicle velocities in rotating body axes. We haven't calculated these yet, so we'll use the standard sim trick of using the answers from the previous time round the sim loop (the answers from the previous time step).

Now, we're going to use equation (2.8), but in reverse. This is because we used non-rotating airspeed vectors *Vu* and *Vv* to work out the aerodynamic forces, so our accelerations are actually already in non-rotating axes.
But we want to use the accelerations to work out velocity vectors *Vi* and *Vj* and those need to be in rotating axes, so we first have to convert the accelerations to rotating body axes. Equation (2.8) in reverse is simply got by subtracting the last term instead of adding it:

$$\begin{bmatrix} Ai \\ Aj \\ 0 \end{bmatrix}_{rotating\ axes} = \begin{bmatrix} Ai \\ Aj \\ 0 \end{bmatrix}_{non-rotating\ axes} - \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}k \end{bmatrix} \times \begin{bmatrix} Vi \\ Vj \\ 0 \end{bmatrix} \qquad (2.9)$$

Working out the vector product term in (2.9) gives the corrections we need to apply:

rotating axes axial acceleration = axial acceleration + $Vj\dot{\theta}$ (2.10)
rotating axes normal acceleration = normal acceleration - $Vi\dot{\theta}$ (2.11)

Will we have to apply the Coriolis theorem to convert our pitching moment into rotating axes? Yes we would, but very fortunately because we're only rotating in one axis (pitch) then all the Coriolis corrections go to zero and can be ignored!

So the pitch acceleration is simply: $pitch\ acceleration = \frac{pitching\ moment}{Iyy}$ (2.12)
where *Iyy* is the moment of inertia in the pitch axis, which will change slowly as mass is ejected out of the nozzle (see appendix 1).

Integration

Right, we now have the accelerations in rotating body axes, so we can integrate them to get the rotating axes velocities:

$$Vi = \int axial\ acceleration\ dt \qquad (2.13)$$
$$Vj = \int normal\ acceleration\ dt \qquad (2.14)$$
$$\dot{\theta} = \int pitch\ acceleration\ dt \qquad (2.15)$$

Which method of numerical integration you use is up to you. You could use simple integration, or you can go the whole hog and use Runge-Kutta integration. The advantage of Runge-Kutta is that it's much more stable: simple integration will explode under the numerical shock of opening a parachute, in which case stop the sim just before opening the 'chute and run another simple sim which descends the vehicle on its 'chute at constant (terminal) velocity.

Now we want to integrate velocity to get distance travelled (displacement). But displacements need to be in non-rotating flat-Earth axes (up, along).
Coriolis again? Yes and no: the velocities we need are the velocities of the vehicle CG. The CG is the centre of rotation, for which the Coriolis corrections for velocity drop down to zero and can be ignored.

However, the velocities of the forebody, tailfins and boatail are not at the CG and so need Coriolis corrections. Fortunately, we've already included these corrections in the numerators of equations (1.7)

So for the vehicle CG:

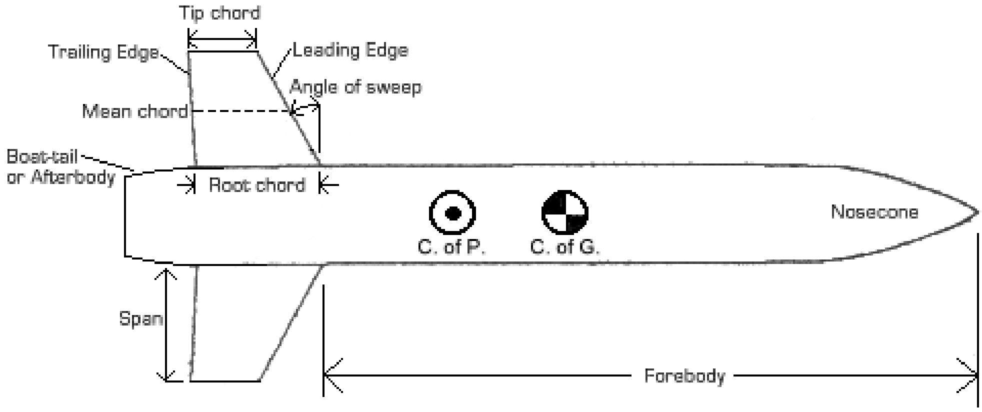
$$Valong = Vi\cos\theta - Vj\sin\theta \qquad (2.16)$$
$$Vup \quad = Vi\sin\theta + Vj\cos\theta \qquad (2.17)$$

And integrating:

$$along = \int Valong\, dt \qquad (2.18)$$
$$up \quad = \int Vup\, dt \qquad (2.19) \qquad\qquad \theta = \int \dot{\theta}\, dt \qquad (2.20)$$

Glossary:

Geometric definitions:



(strictly, the forebody is everything upstream of the boat-tail when there are no fins present.)

Angle of attack: α (or Angle of Incidence)
This is usually referred to as 'alpha', and corresponds to the angle between the airflow direction (usually the **Freestream** direction) and some vehicle or fin datum.

Vehicle: (the)
A stationary object immersed in a moving airflow, or an object moving through stationary air. (Aerodynamically, these two situations are identical in every respect.)
Here, the vehicle is a rocket-vehicle.

Forebody:
The nose and forward fuselage of the vehicle. At subsonic speeds only the nose generates lift, but at supersonic speeds the forward fuselage also generates lift.

Freestream (flowfield): ∞
The undisturbed airflow at a large ('infinite') upstream distance ahead of the **vehicle.**
Freestream properties have the subscript ∞, and are those of the atmosphere.

References:

Ref.1: Mathematical theory of rocket flight, Rosser, Newton, Gross, McGraw-Hill Inc 1947
(now available again in reprint from Amazon)

Ref.2: Stinton, The design of the aeroplane.

Ref.3: Aeronautics and Astronautics Honours Degree course notes.

Ref.4: Rocketry static stability paper
James S. Barrowman, Centuri engineering company inc.

Ref.5: Rocket Services technical report
'Calculation of the centre-of-pressure of a rocket', 1988

Ref. 6: National Association of Rocketry NAR Tech. Report TR-201
'Fundamentals of dynamic stability' by Gordon K. Mandell
Fuselage cross sectional area is denoted here as (A) instead of the original Ar
as this can be confused with Aspect Ratio.

Note that this analysis is described in a series of articles in Apogee newsletters 195-198
(http://www.apogeerockets.com/education/newsletter_archive.asp)

Ref. 7: High power rocketry magazine March 1988
'Wind caused instability' Bob Dahlquist

Ref. 8 'A design procedure for maximising altitude performance' Edward D. LaBudde
Submitted at NARAM 1999

Appendix 1: centre of gravity and pitch moment of inertia

Both of these will slowly change as the rocket burns and ejects mass out of its nozzle.

Starting with the centre of gravity (CG), which is the centroid of mass:
Measure the balance point of the vehicle with the rocket installed but with no propellant in it. This is the CG of the empty vehicle (CG_{empty}), measured from the tip of the nosecone, and the vehicle mass in this condition is $mass_{empty}$
Assume that the CG of the propellant on its own acts half-way along the propellant grain length. Measure the position of this point from the tip of the nosecone. This measurement is then CG_{fuel}, and the mass of the propellant is $mass_{fuel}$

The mass moment about the nosecone tip is then:

$$mass\ moment = mass_{empty}CG_{empty} + mass_{fuel}CG_{fuel} \qquad (3.1)$$

And the overall vehicle CG is found by dividing this moment by the total mass:

$$CG = \frac{mass_{empty}CG_{empty} + mass_{fuel}CG_{fuel}}{mass_{empty} + mass_{fuel}} \qquad (3.2)$$

For hybrids, include a term for the mass and CG of the liquid (assume a nitrous density of 822 kg/m^3)
The overall CG will move with time; simply reduce $mass_{fuel}$ by the amount: nozzle mass flow rate times the time step, for each subsequent time iteration.

Regarding the pitch moment of inertia, this can be estimated by measuring the separate CG's of each component of the rocket, and summing them together:

pitch moment of inertia $I = \sum(mr^2)$ (3.3)

where the *r*'s are the individual component CG positions measured relative to the overall vehicle CG.

But this is tedious, and some components such as the body tubes have to have their moments of inertia especially calculated as they're not 'point masses' (see equ. (3.6) below). Furthermore, the overall CG position moves as propellant is ejected out the nozzle.

It's much easier simply to measure the pitch inertia of the empty vehicle (no propellant).
This is done by suspending the vehicle on the points of a pair of pins and letting the vehicle swing as a pendulum. (Don't use large swings, small swings are more accurate).
Measure the time taken for a swing (measure 10 swings and divide by 10 for accuracy).
The moment of inertia about the pins pivot point is then found from the equation for a physical pendulum:

$$I_{pins} = \frac{\tau^2 mgh}{4\pi^2} \qquad (3.4)$$

Where τ is the period of each swing in seconds, *m* is the vehicle mass, *g* is 9.81 m/sec^2 and *h* is the distance between the pivot pins and the vehicle's CG.
This gives the moment of inertia about the pins pivot, but you have to convert this to get the moment of inertia about the vehicle CG.

This can be done using the Parallel axes theorem, which states:

$$I_{CG} = I_{elsewhere} - mr^2 \qquad (3.5)$$

Where *m* is the vehicle mass, and *r* is the distance between the CG and the 'elsewhere', in this case the pins pivot point.
You can then calculate the moment of inertia of the propellant about its own CG, and then use equ. (3.5) to re-reference it to the vehicle (with propellant loaded) CG. Then simply add this moment of inertia to the empty vehicle moment of inertia.
The moment of inertia of a thick-walled tube of propellant about its own CG (at half its length) is:

$$I_{fuel} = m\frac{R^2+r^2}{4} + m\frac{l^2}{12} \qquad (3.6)$$

Where *R* is the outer radius of the tube (metres), *r* is the inner radius, *l* is the tube length, and *m* is its mass. Remember that *r* will increase with time as fuel is burnt off the inner surface of the propellant.

Calculating the moment of inertia of the nitrous oxide in the run tank of a hybrid is rather complicated, but the resulting equations are given in our paper 'Hybrid effects on stability'.

Appendix 2: derivation of the jet damping term

The thrust of a rocket relies on the physical principle that the total linear momentum of the system comprising the vehicle plus the exhaust mass is conserved.
But the total angular momentum of this system is also conserved, which causes a sizable effect on the vehicle when it rotates in pitch during motor firing. It causes a sizable damping moment called jet damping, an extremely useful side-effect.

Firstly, the moment of momentum of the rocket vehicle of mass *m* is: $mk^2\dot{\theta}$ (4.1)
where *k* is the pitch axis radius of gyration.

(Recall that the pitch Moment of inertia I = $\sum(mr^2) \Rightarrow (\sum m)\, k^2$ (4.2)

We assume that the rocket nozzle is alligned with the long axis of the vehicle.
Thus the sidewise component of velocity of the gas issuing from the nozzle is just:
$v = r_e\dot{\theta}$ (4.3)

where r_e is the distance from the center of gravity of the rocket to the exit of the nozzle.

Then using (4.3), the nozzle is removing moment of momentum (*H*) from the vehicle at the rate:

$$\frac{dH}{dt} = \frac{d}{dt}(r_e \times mv) = \dot{m}r_e^2\dot{\theta} \quad (4.4)$$

So the rotation of the vehicle about the (constantly shifting) Centre of Gravity in response to an applied mostly-aerodynamic total moment *M* is described by:

$$M = \frac{dH}{dt} = \frac{d}{dt}(mk^2\dot{\theta}) + \dot{m}r_e^2\dot{\theta} \quad (4.5)$$

Differentiating the bracketed term of (4.5), we get:

$$\frac{d}{dt}(mk^2\dot{\theta}) = -\dot{m}k^2\dot{\theta} + m\dot{\theta}\frac{d}{dt}(k^2) + mk^2\ddot{\theta} \quad (4.6)$$

If we are only looking at short time periods such as a few pitch oscillations, we can then justify dropping the middle term of (4.6) involving the change of the radius of gyration with time:

$$m\dot{\theta}\frac{d}{dt}(k^2)$$

We do so, and (4.5) becomes:

$$mk^2\ddot{\theta} = -\dot{m}\dot{\theta}(r_e^2 - k^2) - M \quad (4.7)$$

Since *m*, $\dot{m}$, k^2, and $(r_e^2 - k^2)$ are positive, the term $\dot{m}\dot{\theta}(r_e^2 - k^2)$ in (3.7) tends to give $\ddot{\theta}$ a different sign from $\dot{\theta}$. That is, the effect of the term $\dot{m}\dot{\theta}(r_e^2 - k^2)$ is to decrease the absolute value of $\dot{\theta}$, or in other words to slow up the rate of rotation.
For this reason, the term $\dot{m}\dot{\theta}(r_e^2 - k^2)$ is referred to as a "*jet damping term*." It has the effect of helping to damp out any pitching motion.

Most standard texts dump the k^2 term; so the jet damping moment equation is usually simplified to:

$$Jet\ damping = -\dot{m}\dot{\theta}\, r_e^2 \qquad (4.8)$$

Where $\dot{m}$ is the mass flow rate of mass out the nozzle in kg/second.