Feed Forward 2.0

Betaflight 4.1 brings us `Feed Forward 2.0` - a comprehensive set of updates to FeedForward.

Note that `ff_spread` from the development versions has been changed to `set ff_interpolate_sp = AVERAGED`. The time interval to spread is no longer required.

What is FeedForward?

FeedForward (FF) is a drive factor that increases stick responsiveness. It is proportional to the instantaneous derivative, or 'rate of change' of stick movement. The faster the sticks are moving, the more FeedForward we get. FeedForward helps P drive the quad into turns. Unlike P, FeedForward cannot cause oscillation, no matter how much FF is added.

With FeedForward, we get better stick responsiveness without pushing P so high as to cause wobbles. It also reduces the delay time between input and response. Less delay means less error and less I windup/overshoot. It is great for racing, LOS and radical freestyle flying. It isn't great for cinematic HD.

Too much FeedForward can cause:

- overshoot at the start of a flip, particularly when the sticks hit 100% travel
- exaggerated responsiveness to RC steps
- amplification of shaking when the pilot is cold or nervous
- spikes in the motor traces and brief shakes with big RC steps
- gyro getting ahead of setpoint

Since RC data comes into the flight controller in 'packets', we actually get a series of sudden steps coming in. The derivative of an instant step is an infinitely high spike, which is basically useless. Back in 3.4.0 we introduced 'filter' mode RC smoothing, where an 'input' lowpass filter rounded off those steps; the derivative now became a wider spike that we could use. We then applied a 'derivative' lowpass over that spike, which smoothed it out into a kind of 'lump' of FF for each incoming step. AUTO RC filtering adapts the smoothing filters dynamically to the apparent

incoming RC step rate; manual setups can be fine-tuned to any desired degree of smoothness, at the cost of input delay.

Currently erratic RC steps are a major problem. OpenTx 2.3 improves on this a bit for FrSky users. TBS are introducing a 'locked 150hz mode' that may be better.

4.1 introduces several novel `ff_2.0` technologies that improve on these limitations.

ff_boost

This is a fantastic new feature that markedly reduces delay in fast setpoint changes.

Most motors take time to spin up / slow down. They need to be pushed harder at the very start of a move than in the middle. FF and P both ramp up slowly at the start of a move because initially our fingers move slowly and most pilots use Expo in their Rates setup. So until now, the motors haven't got that immediate push needed unless really high FF gains were used (250+). But with FF gains of that magnitude, overshoot is hard to control, especially because of SuperRate being applied at the end of the stick travel, really causing FF to boost the move right when we need the quad to start to slow down to not overshoot.

'ff_boost' is a PID parameter that is proportional to the stick acceleration. Technically it is the second derivative of Setpoint.

Being the second derivative, `ff_boost` peaks really early in a move, the instant the sticks start moving. It then eases away to zero when the sticks are at mid-travel, because at constant stick velocity we have lots of FF, but no acceleration. ff_boost finally goes negative as the sticks slow down.

The acceleration component does exactly what we need to help overcome motor delay - pushes early, eases off mid-move, and actively slows down the motors at the end. With the right amount of FF and ff_boost, a responsive quad can have nearly completely lag-free tracking of inputs and not get overshoot.

Typically `set ff_boost = 15`, the default, is about right for most quads. Larger values may be used on low-authority quads or quads with motors that are slow to spool up. More than 40 would be unusual.

The strength of the boost is directly linked to the amount of FF set in the PIDs. Think of ff_boost as a 'factor' that modulates the timing of how your FF gets applied to the motors. More boost brings the FF on earlier.

Advanced users who can log should do a test flight with fast twitchy inputs, quick left/right stabs, and look for the timing and amount of their FF signal. Ideally P

should have little work left to do when the boost/FF components are correct.

ff_boost can work without any other of the ff_2.0 features being active.

ff_interpolate_sp

This is a 'digital' way of calculating FF from each new RC 'step'. It gives a cleaner feedforward trace than with the old 'filter' method, with less delay.

`set ff_interpolate_sp = ON` analyzes each new incoming RC data packet, and the calculated change in setpoint is converted to an immediate step up in FF. Each step up is held constant until the next RC data step arrives.

The sharp corner of the FF step can be smoothed according to the rc_smoothing_derivative lowpass filter frequency. This is set dynamically by default in AUTO mode, but can be manually overridden if desired. 120hz is a good smoothing value for a clean feed-forward trace. 20hz smooths out traces with a fair bit of up and down; 10hz may be needed for very long range and cinematic quads where the large steps from 50hz mode and poor quality links can make the quad jerky in turns.

Dropped RC packets will normally cause sudden FF drops to zero. `set ff_interpolate_sp = AVERAGE` is intended to help manage this specific issue. Alternatively, a low rc_smoothing_derivative filtering value can smooth the drops out a bit, at the cost of incoming RC delay.

ff_max_rate_limit

ff_max_rate_limit attenuates / prevents FF induced overshoot at the start of a flip.

When performing quick flips or rolls, the sticks typically stop suddenly when they hit the physical limit of their travel. With Expo and SuperRate applied to your Rates, the rate of increase in Setpoint is greatest just before the sticks stop. At this point in time, the quad itself is turning really fast and has a ton of rotational momentum. The Expo\SuperRate effect means FF is absolutely massive too. All this, just before the sticks suddenly come to a grinding halt.

Significant overshoot, even with a lot of D, is basically inevitable. In previous versions, you could use D_min to boost your D-gains high enough to compensate, but it was difficult to find the balance. Additionally, the transition would be sudden and the opposing pair of motors have to spin up and go full throttle, which in turn causes wobbles on the other axes and makes the quad climb. Using a ton of D to control this has its own problems, eg noise and dumbing down responsiveness at other times.

ff_max_rate_limit predictively identifies situations in which the sticks are likely to hit their limit, and cuts FF in anticipation of that happening. It kind of 'looks ahead' and pre-emptively reduces FF just at the right time, typically totally eliminating the overshoot that would otherwise happen.

A major benefit is that it markedly reduces the need for the opposing motors to spin up. Flips become cleaner and more accurate than before, and climb on flipping under LOS conditions is much less.

The default value of 100 works well, and attempts to reach FF outputs of 0 as gyro values reach setpoint. Lower values result in feedforward influence tapering to 0 at a percentage of those values (e.g. at a value of 50, in a sustained flip, once gyro reaches half the setpoint value the feedforward term will taper to zero). Higher values allow for feedforward to continue influencing the PID controller. For higher authority craft this will cause overshoot, but for authority-limited craft these higher values can be very beneficial. ff_max_rate_limit is not active as the sticks return to center.

To determine ff_max_rate_limit and ff_boost works best, look at the start of a hard flip in BlackBox Explorer and see if there is any overshoot. If with ff_max_rate_limit = 100 there is still too much overshoot, first evaluate if Feedforward or P-term is driving overshoot at the time of interest. If so, try ff_max_rate_limit = 95. If the overshoot is too well controlled, try 105 to 110. The range of adjustment is quite tight.

ff_interpolate_sp = AVERAGED

Since FF is calculated for each incoming RC step, we have a problem when a new step doesn't arrive as it should.

Usually this is caused by 'dropped packets', or when a TBS or R9 Rx switches out of 150hz mode to the slower 50hz mode. Anytime this happens, FF would normally drop suddenly to zero; when the next valid data value comes in, a double-height step up occurs. These zero/double-height pairs in the FF trace are very messy.

In `AVERAGED` mode, the interpolation algorithm averages each two successive feed forward values. In the case of a sudden drop to zero with a large subsequent step up, this is changed to a smaller drop, a middle value, and a jump up. Overall the steps are smaller.

There is a downside to using `AVERAGED`. If the pilot makes a fast input, then suddenly holds the sticks perfectly still, ff_spread will hold FF at the previous, high, value for the set time, rather than dropping to zero immediately as it should. This can cause

an overshoot for the duration set by the extend value. It is most easily seen at the start of a flip when the sticks hit their physical limit, but can happen at other times.

`AVERAGED` is most useful for long-range / cinematic quads with substantial RC smoothing.

For racers and general use the quality of the RC link depends mostly on how good your antennas are and how far you plan to fly. Close-in racing with good antennas, especially with Futaba and some Spectrum radios that have very consistent RC steps, can work best without averaging.

For technical details and more info,check the original pull request #8623