POWER4FLIGHT

Getting the most from IntelliJect

Bill Vaglienti
Power4Flight
202 Wasco Loop, Suite 104
Hood River, OR 97031
541-308-0650

April 2, 2024

IntelliJect is more than a simple fuel injection system. It was designed for autonomous aerospace applications and contains features that can improve the performance and reliability of aerospace systems. This document highlights some of the more important, but sometimes overlooked, features of IntelliJect that can improve your system. Refer to the IntelliJect User's Manual for details on all features of the IntelliJect fuel injection system.

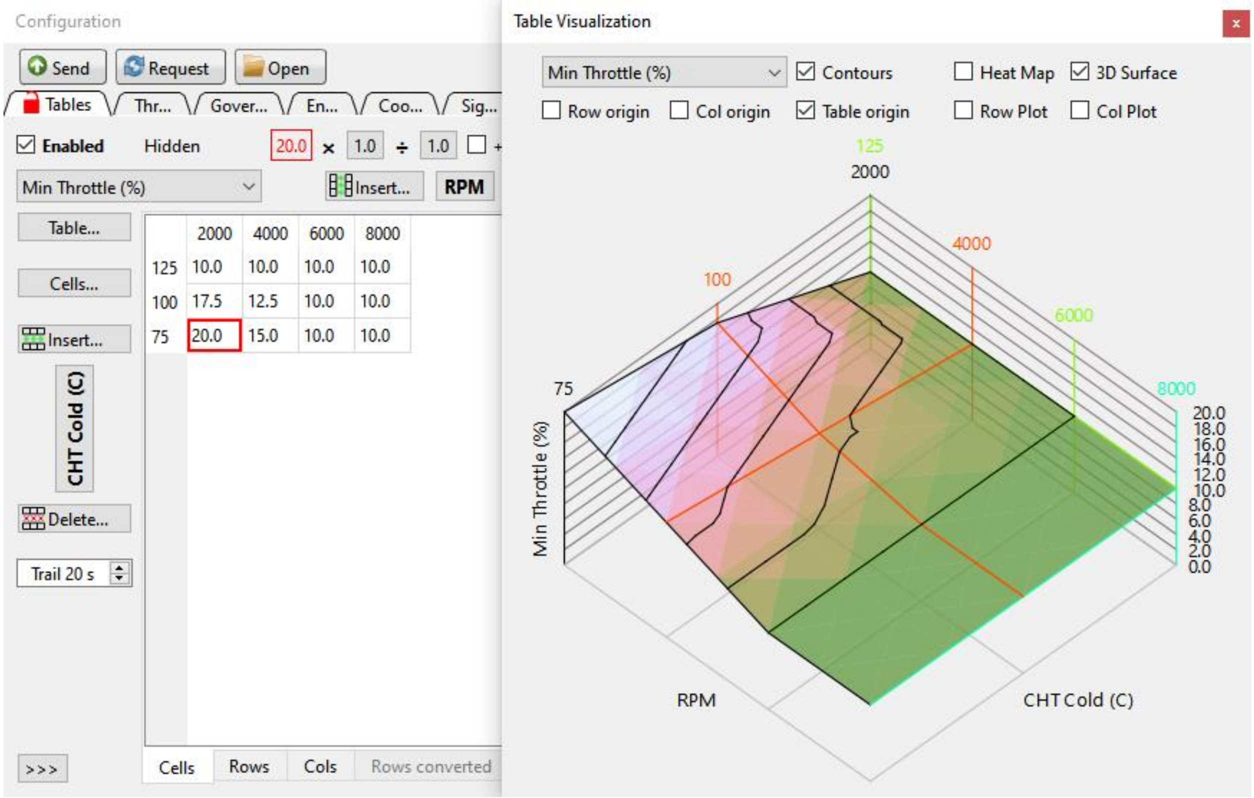
I. THROTTLE CONTROL

Perhaps the most useful feature of IntelliJect is the ability to separate the throttle command from the throttle control. Throttle command can be received via communications (serial or CAN) or a PWM input signal. The throttle output can be sent over CAN to a servo, or via a PWM output to a servo. The ability to interpret the commanded throttle allows for advanced behaviors:

- Enforce minimum and/or maximum throttle limits.
- Limit the rate of change of the throttle.
- Curve the throttle to linearize the engine power response versus command.
- Enforce RPM limits or allow for direct RPM command.

A. Minimum Throttle

Minimum throttle is determined from two tables: the limit comes from the table that has the largest minimum throttle. If the engine is cold, or the air density is thin, you can use these tables to prevent the throttle going low enough to stall the engine. An example of two minimum throttle tables that encode these behaviors is below.



☑ Enabled 20.0 × 1.0 ÷ 1.0

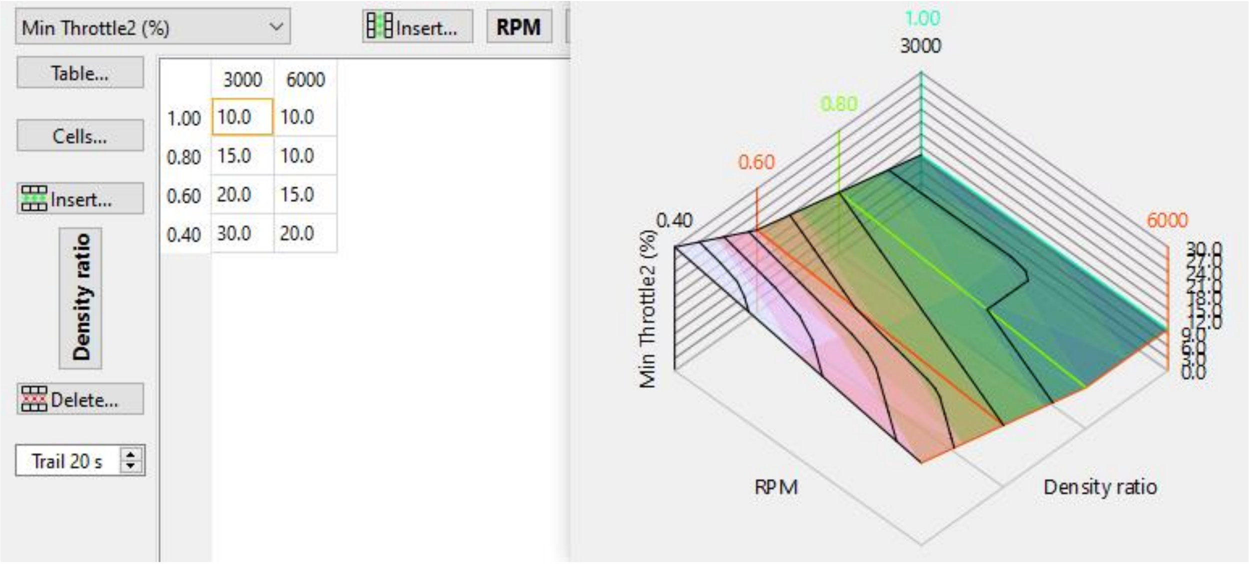
Min Throttle (%)

	2000	4000	6000	8000
125	10.0	10.0	10.0	10.0
100	17.5	12.5	10.0	10.0
75	20.0	15.0	10.0	10.0

Min Throttle (%) ☑ Contours ☐ Heat Map ☑ 3D Surface

☐ Row origin ☐ Col origin ☑ Table origin ☐ Row Plot ☐ Col Plot

Figure 1. First minimum throttle table – prevent stalling due to cold temperatures.



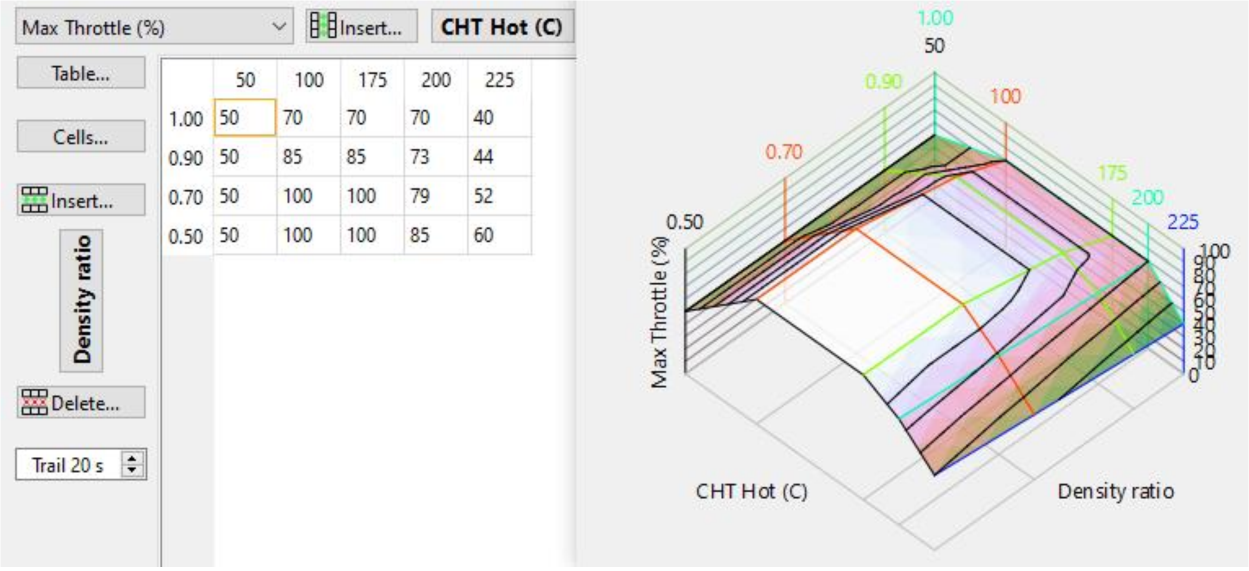
	3000	6000
1.00	10.0	10.0
0.80	15.0	10.0
0.60	20.0	15.0
0.40	30.0	20.0

Figure 2. Second minimum throttle table – prevent stalling due to low air density.

B. Maximum Throttle

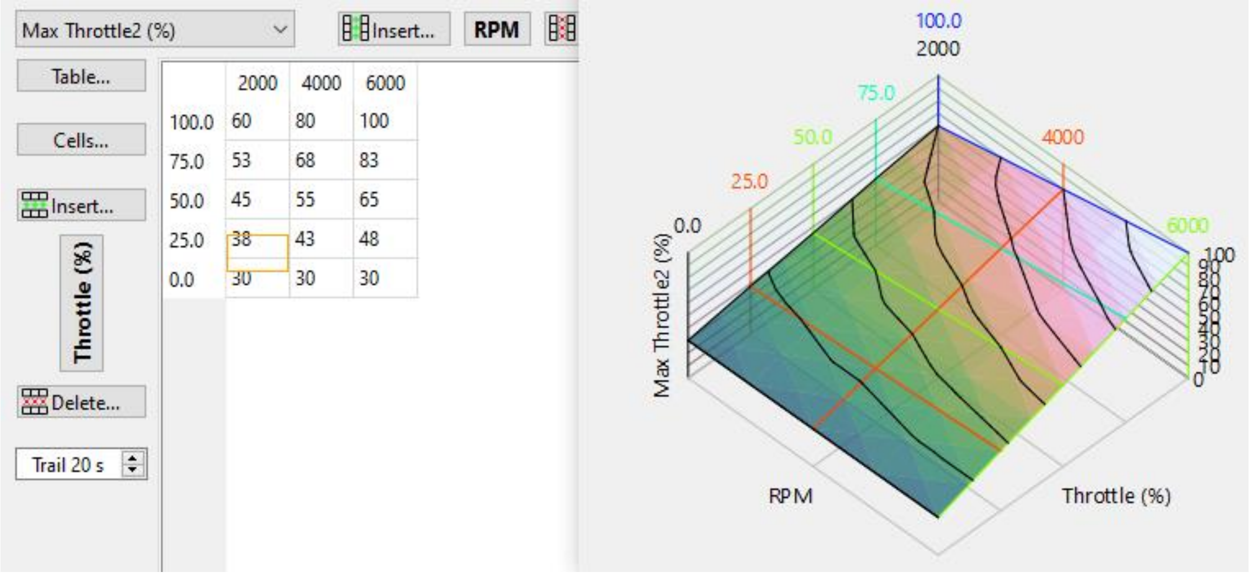
Like the minimum throttle the maximum throttle can be set using two tables. When setting these tables consider:

- If the engine is cold limit the throttle because high loads and cold temperatures could cause thermal stress to the engine.
- If the engine is hot, limit the throttle to prevent catastrophic overheating.
- If the air density is high, you may want to limit the throttle to prevent knocking.
- Knocking can be further mitigated by limiting the low-speed throttle, only allowing the engine to reach full throttle output at high engine speeds.



	50	100	175	200	225
1.00	50	70	70	70	40
0.90	50	85	85	73	44
0.70	50	100	100	79	52
0.50	50	100	100	85	60

Figure 3. Maximum throttle table – limit throttle for air density or overheating.



	2000	4000	6000
100.0	60	80	100
75.0	53	68	83
50.0	45	55	65
25.0	38	43	48
0.0	30	30	30

Figure 4. Second maximum throttle table – prevent high load at low speed.

C. Throttle Curve

The power produced depends on the throttle position and engine speed. However, even increments of throttle position do not result in even increments of engine power. In most applications the throttle is commanded by a flight control system whose control laws will perform much better if the throttle to power transfer function is linear. Use the throttle curve feature to linearize the response. This is easily done for fixed pitch propeller applications using tools built into IntelliJect Display.

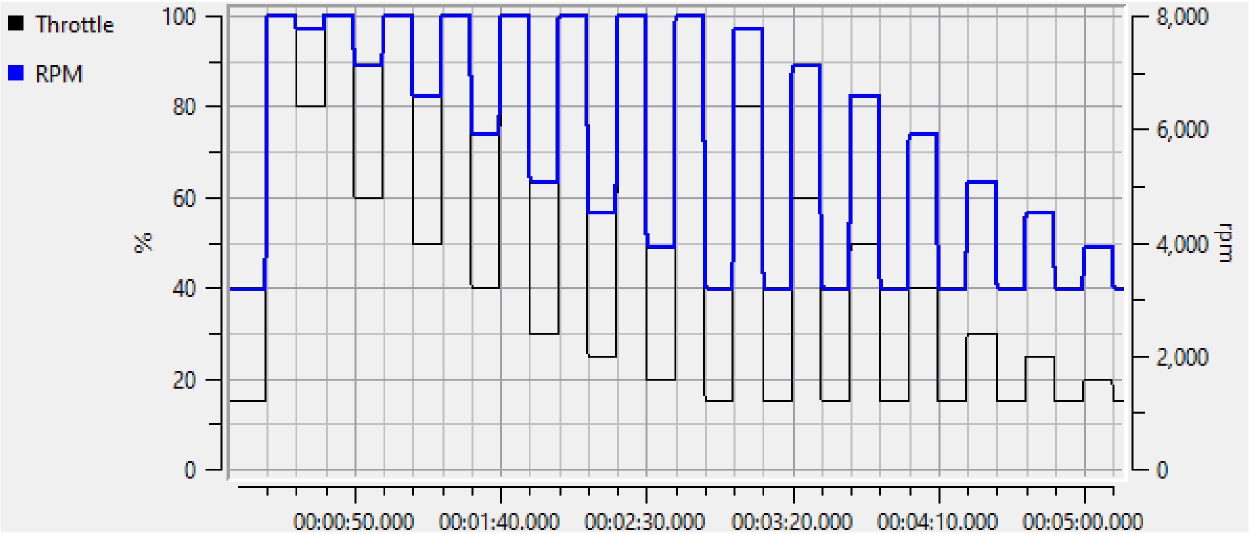
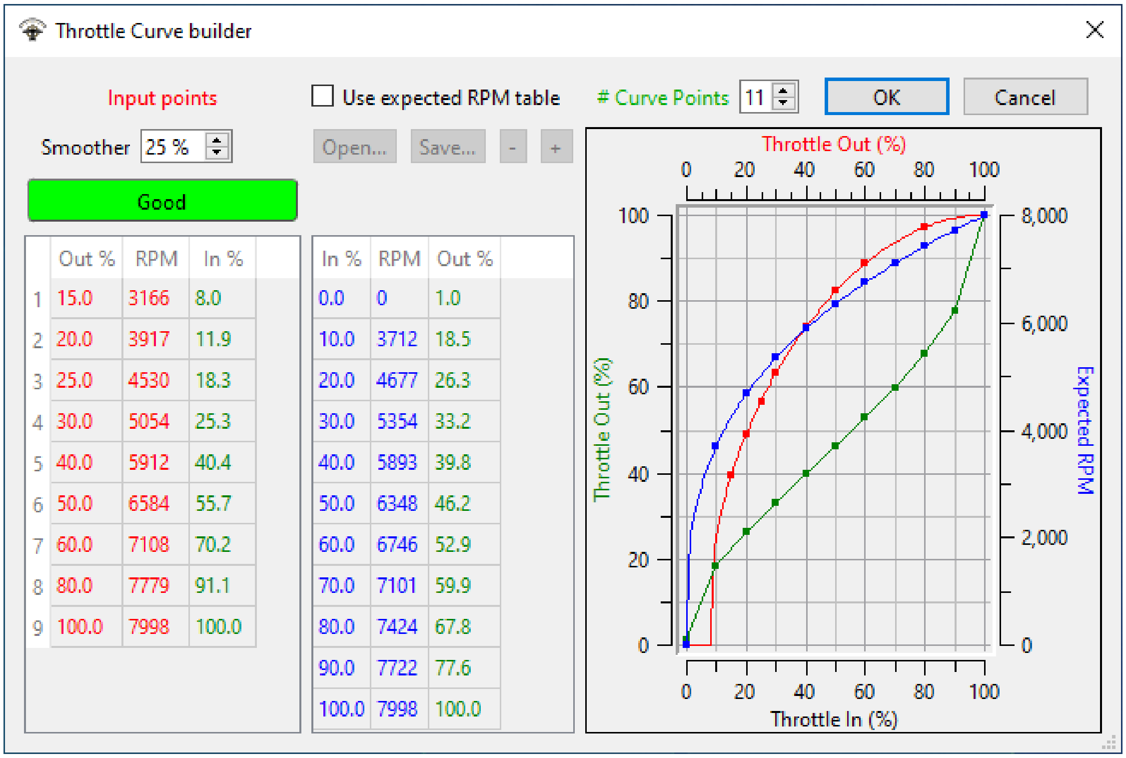


Figure 5. Chart of engine test data used to compute the linearized throttle curve.

The resulting throttle curve will not only improve the flight control performance, but it will also be used by IntelliJect's own feedback control laws that govern the engine RPM. You can also use the curve to harmonize differences between engines – so the flight controller can be tuned for a single engine response curve, even if the engine hardware changes.



☐ Use expected RPM table

Curve Points 11

Smoother 25 %

	Out %	RPM	In %
1	15.0	3166	8.0
2	20.0	3917	11.9
3	25.0	4530	18.3
4	30.0	5054	25.3
5	40.0	5912	40.4
6	50.0	6584	55.7
7	60.0	7108	70.2
8	80.0	7779	91.1
9	100.0	7998	100.0

In %	RPM	Out %
0.0	0	1.0
10.0	3712	18.5
20.0	4677	26.3
30.0	5354	33.2
40.0	5893	39.8
50.0	6348	46.2
60.0	6746	52.9
70.0	7101	59.9
80.0	7424	67.8
90.0	7722	77.6
100.0	7998	100.0

Figure 6. The throttle curve built from the test data – linearizing the cube of RPM.

D. Low and High RPM limiter

IntelliJect provides the option for direct RPM control, which engages a control law that moves the throttle to achieve a desired RPM. However, this control law can only be used if the flight controller can generate an RPM command. In most cases the flight controller would prefer to generate a throttle command. You can still use the RPM controller in those cases:

- Turn on the "Throttle commands RPM" feature, which will convert the throttle command to an RPM command. See the IntelliJect User's manual for more details. -OR-
- Turn on the "Low RPM limiter" and/or the "High RPM limiter".

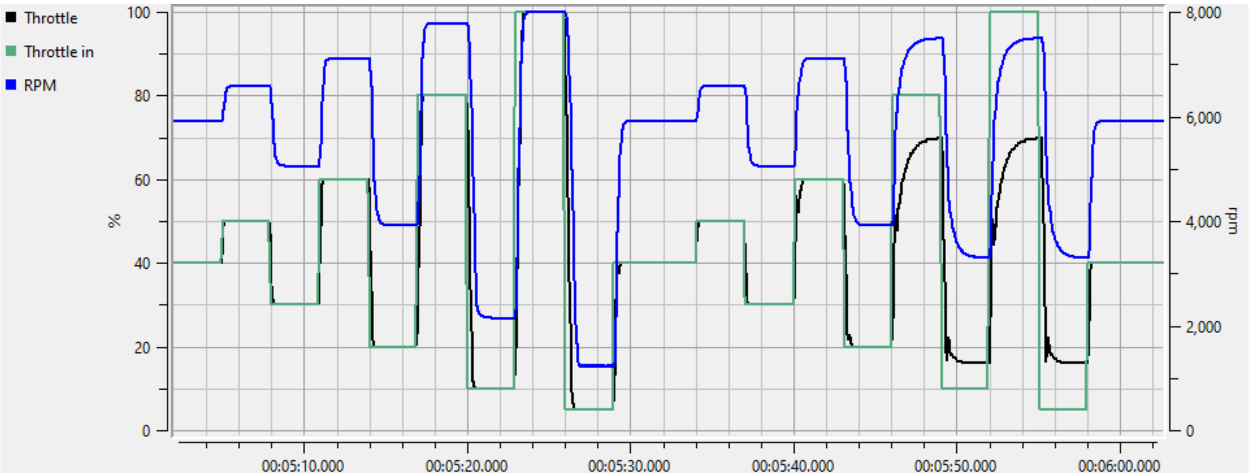
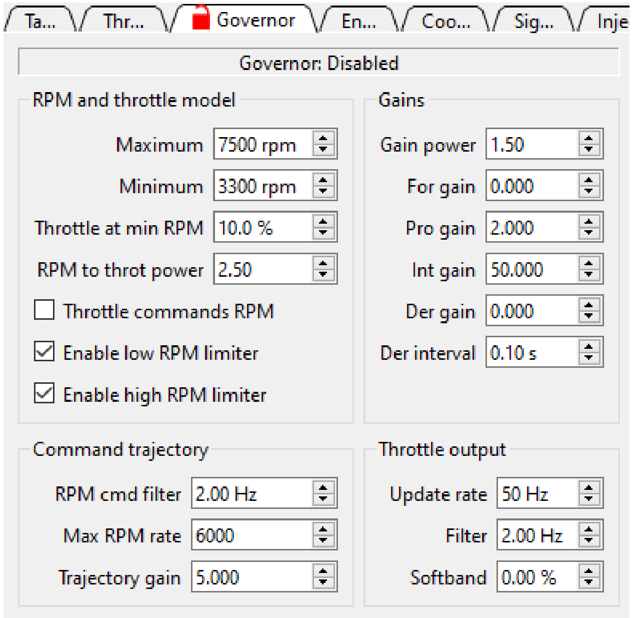


Figure 7. Rapid throttle changes without and with the rpm limiter engaged.

The RPM limiter works by overriding the throttle command if the engine is on its way to violating the minimum or maximum RPM. Enabling the limiter can allow you to relax the throttle limits a bit – letting the engine idle lower or get closer to redline.

The limiters work using the same feedback strategy as the RPM controller (in trajectory mode) so the controller must be well tuned. See the IntelliJect User's manual for details.

In Figure 7 the limiters were set to 3300 and 7500. You can see that throttle commands which would not violate the limits are not significantly affected, but large throttle commands are overridden to keep the RPM from exceeding the limits. For illustration purposes in this example the throttle was allowed to move at 200%/sec, and the throttle limits were turned off. Normally you would choose a slower throttle rate and have more conservative throttle limits.



Governor: Disabled

RPM and throttle model

Maximum 7500 rpm

Minimum 3300 rpm

Throttle at min RPM 10.0 %

RPM to throt power 2.50

☐ Throttle commands RPM

☑ Enable low RPM limiter

☑ Enable high RPM limiter

Gains

Gain power 1.50

For gain 0.000

Pro gain 2.000

Int gain 50.000

Der gain 0.000

Der interval 0.10 s

Command trajectory

RPM cmd filter 2.00 Hz

Max RPM rate 6000

Trajectory gain 5.000

Throttle output

Update rate 50 Hz

Filter 2.00 Hz

Softband 0.00 %

Figure 8. Enabling rpm limiters.

II. FUEL INJECTORS

Unlike large automotive engines with oxygen sensors and carefully sized injectors, the typical small aerospace engine does not have mixture feedback and must work with off the shelf injectors that are not sized optimally for the engine. IntelliJect has multiple features to manage the idiosyncrasies of the injectors. Using these features improves reliability and accuracy.

A. Injector Characterization

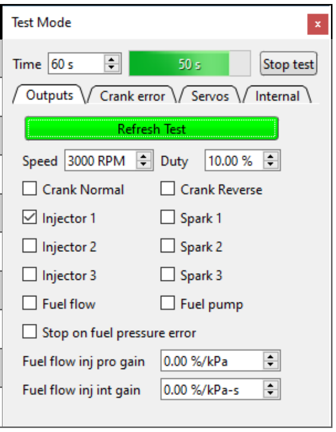
IntelliJect works by computing the desired mass of fuel to inject. The fuel mass is used to compute the injector opening time, which depends on three terms called the injector characterization:

- The flow rate through the injector when it is held open, at the designed fuel rail pressure.
- The injector trim: the amount of time needed to open the injector before any fuel flows.
- The injector minimum opening time, which is the amount of time below which the injector output is not linear.

All three injector terms can be determined by measuring fuel delivered as a function of the injector opening time. You can use the IntelliJect test mode feature, in combination with an independent fuel flow meter, to make the measurements. In this example IntelliJect operated the injector for 60 seconds at various duty cycles (while maintaining fuel pressure). For each test the fuel delivered was measured, and the pulse width and fuel per pulse are computed.

Table 1. Injector calibration data.

Ne (RPM)	Δt (s)	Duty (%)	Fuel (g)	PW (ms)	FF/pulse (mg)
~~3000~~	~~60~~	~~3~~	~~0.00~~	~~0.60~~	~~0.00~~
~~3000~~	~~60~~	~~4~~	~~0.20~~	~~0.80~~	~~0.07~~
3000	60	5	1.55	1.00	0.52
3000	60	7	2.88	1.40	0.96
3000	60	10	4.71	2.00	1.57
3000	60	15	7.54	3.00	2.51
3000	60	20	10.38	4.00	3.46
3000	60	30	16.18	6.00	5.39
3000	60	50	28.34	10.0	9.45
3000	60	75	43.30	15.0	14.43



Test Mode

Time 60 s

50 s

Speed 3000 RPM

Duty 10.00 %

☐ Crank Normal

☐ Crank Reverse

☑ Injector 1

☐ Spark 1

☐ Injector 2

☐ Spark 2

☐ Injector 3

☐ Spark 3

☐ Fuel flow

☐ Fuel pump

☐ Stop on fuel pressure error

Fuel flow inj pro gain 0.00 %/kPa

Fuel flow inj int gain 0.00 %/kPa-s

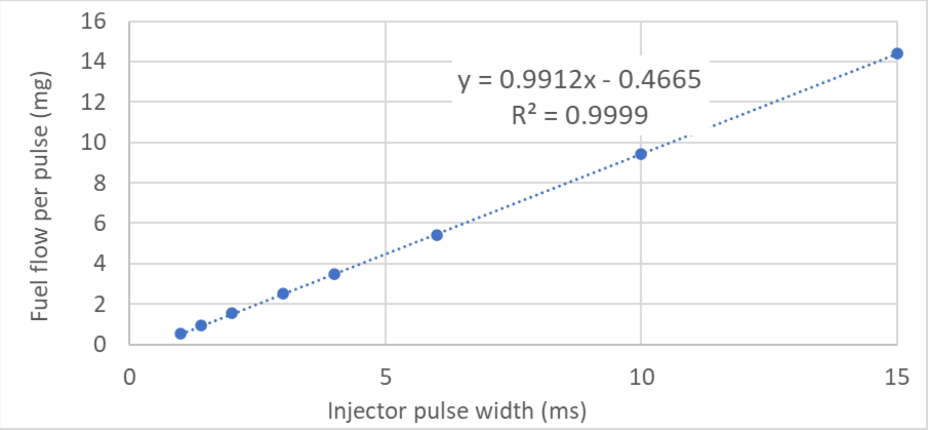
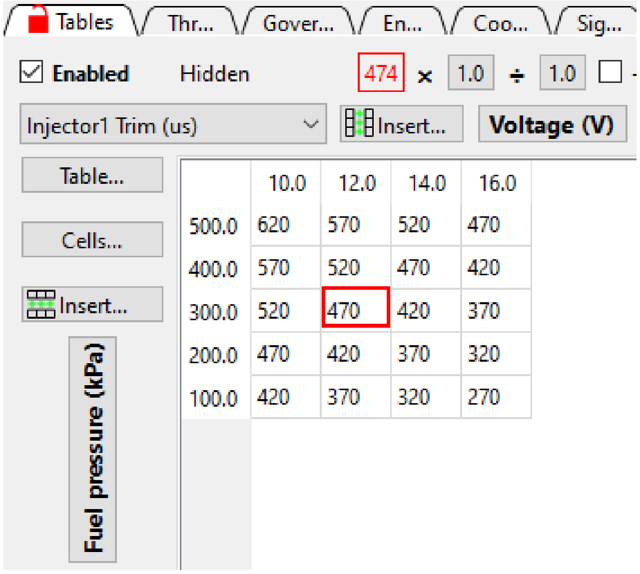


Figure 9. Plotted injector calibration data.

The slope in Figure 9 is 0.9912 milligrams of fuel per millisecond of opening time, this translates to an injector open flow rate of 0.9912×60 = 59.472 grams per minute. The offset is -.4665 milligrams at zero opening time. This means the opening time for zero fuel (i.e. the injector trim) is .4665/.9912 = 0.470 milliseconds.

Clearly, the injector does not produce negative fuel flow when the opening time is zero; the offset computed above is a result of the injector pintle opening slower than it closes, and it implies that short injector times will have a non-linear response. The curve fit in Figure 9 does not use the first two points of the calibration data because of this effect. This leads to the third injector characterization term: the minimum opening time. Opening this injector for less than 1 millisecond will result in non-linear response.



☑ Enabled

474 × 1.0 ÷ 1.0 ☐

Injector1 Trim (us)

Voltage (V)

Fuel pressure (kPa)

	10.0	12.0	14.0	16.0
500.0	620	570	520	470
400.0	570	520	470	420
300.0	520	470	420	370
200.0	470	420	370	320
100.0	420	370	320	270

Figure 10. Example injector trim table.

The injector trim also depends on the voltage of the injector rail, and on the fuel pressure. Each of these variables will change the speed at which the injector pintle moves, affecting the opening versus closing times. Therefore, a complete injector characterization will include trims at different voltages and fuel pressures. Note that the fuel flow rate only needs to be measured at one fuel pressure, IntelliJect will automatically compensate based on the square root of the fuel pressure. Also note that varying voltage is only an issue for versions of IntelliJect that do not include the 12V regulator.

Why should you care?

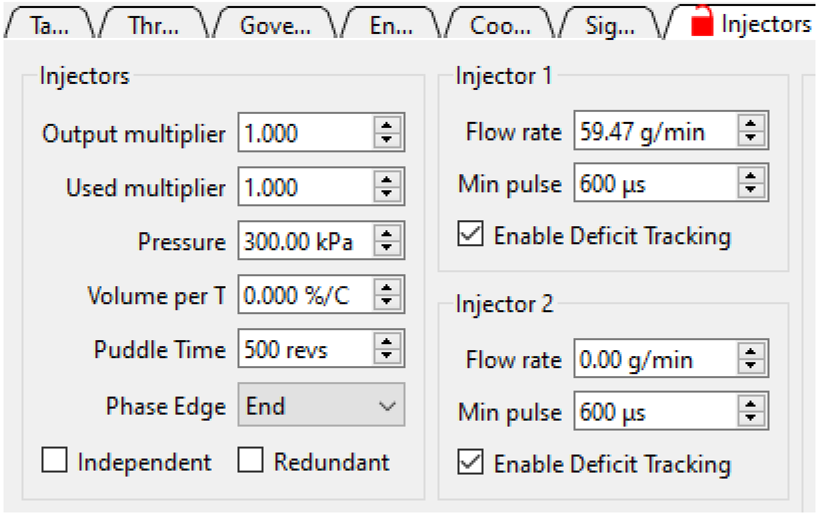
Consider the implications of ignoring the offset term: If an engine used 10.38 grams/minute of fuel at 3000 rpm at sea level, the injector would need to open for 4 milliseconds on each revolution. However, if the engine goes to 22,000 ft the same operating condition will demand half the fuel (the air density is half at 22,000 ft). Without the injector offset IntelliJect would compute an injector opening time of 2 milliseconds, which would not produce the required fuel flow, leading to an engine that was 10% too lean. With the offset IntelliJect would correctly compute that half the fuel would be delivered at an opening time of 2.235ms instead of 2ms. In addition to the correct mixture control, the fuel used estimate is also more accurate if the injector is properly characterized. Both benefits will be highly appreciated by users of your engine.

B. Injector Dynamic Range

Little engines have a wide speed range, and this combined with the dearth of options for small injectors means you may be working with an injector that is too big for the engine. IntelliJect can help with this by specifying the minimum opening time. If the injector time (before the trim is applied) is less than the minimum opening time IntelliJect will simply skip the injector event, adding the skipped injector time to the next injector output. Note that the injector trim is only applied once per injector output. Depending on settings IntelliJect may skip multiple events in succession. For manifold-injected two-stroke engines the skipping is not problematic, as many as five outputs may be skipped before you notice the engine behavior change, as long as the total fuel delivered remains correct (Four-stroke engines cannot tolerate as much skipping).

Utilizing the injector minimum opening time will improve linearity and increase the options available for injector selection. Also consider the altitude case: you may not need to skip outputs at sea level, but as the engine goes to altitude the reduced fuel flows may need skipping to remain accurate.

The final dynamic range consideration is injector deficit tracking.



Injectors

Output multiplier 1.000

Used multiplier 1.000

Pressure 300.00 kPa

Volume per T 0.000 %/C

Puddle Time 500 revs

Phase Edge End

☐ Independent ☐ Redundant

Injector 1

Flow rate 59.47 g/min

Min pulse 600 µs

☑ Enable Deficit Tracking

Injector 2

Flow rate 0.00 g/min

Min pulse 600 µs

☑ Enable Deficit Tracking

Figure 11. Injector minimum time and deficit tracking.

Deficit tracking remembers the difference between what was asked for and what was delivered. There could be a difference due to a temporary loss of fuel pressure, or due to a transient fueling condition that needs more than 100% injector duty. Deficit tracking remembers the difference and applies it to the next injector output event. Deficit tracking even works for negative fuel flows, which may arise due to transient fueling requirements.

III. REDUNDANCY

IntelliJect has several redundancies built in that can improve the reliability of your system. The redundancies are cheap insurance that might keep an engine running well enough to come home, saving an expensive aircraft.

A. Dual crank sensors

The crank sensor is a critical element of any ECU; if it fails your engine will stop. Install two sensors and connect both to IntelliJect. Not only will your engine keep running if a sensor fails (most commonly due to a broken wire) but you will get more benefits:

- With two sensors you can tell IntelliJect which one you prefer, and it will use that sensor unless the sensor has a problem. You can even let IntelliJect choose the preferred sensor, which it will do on a per-output basis, choosing the sensor which is closest in position to the output event.
- The two sensors cross check each other, so if a sensor is having a problem, not only will the engine keep running, but you'll get an error indication telling you which sensor is bad.
- If the two sensors are spaced appropriately IntelliJect will determine the direction of rotation based on the timing of the sensors. The rotation information can be used to prevent the engine from running in the wrong direction; if, for example, your customer attempts to crank the engine in reverse direction.
- When both sensors are working IntelliJect will report the crank angle of sensor #2 as measured by sensor #1. If this value does not match expectations, you know at least one of the sensors is not where you think it should be.
- The sensors can be dissimilar types if desired: a multi-tooth wheel sensor for one input, and a simple once-per-rev sensor for the other – or any combination you can dream up. IntelliJect treats the sensors completely independently.

B. Multiple temperature measurements

IntelliJect has four temperature inputs, configured to measure RTDs or thermistors. Even if you only have a single cylinder, you should use at least two cylinder temperature sensors. If a sensor fails (again, most commonly due to broken wires) IntelliJect will flag the sensor as bad and use the other one.

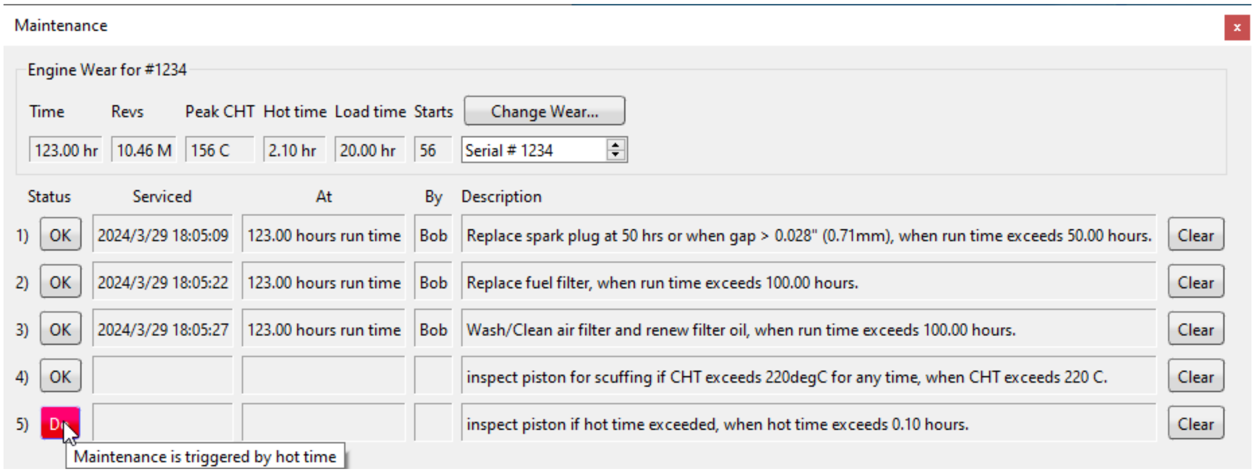
For multi-cylinder engines you can use multiple sensors to affect cylinder-specific cooling control and cylinder-specific injector control. Again, if a sensor fails, IntelliJect will use the good sensor for both cylinders.

IntelliJect keeps track of key “wear” variables:

- The total time the engine has been running.
- The total number of revolutions the engine has made.
- The hottest temperature measurement.
- The amount of time the engine has run while hot.
- The amount of time the engine has run at a high load.
- The number of times the engine has been started.

While these variables are interesting in and of themselves, they are useful for maintenance tracking and electronic log-booking. You can configure a list of maintenance tasks to be performed as a function of these variables. Once the relevant variable has passed the maintenance threshold an error bit is reported to the user. For example, the starter belt might need replacing after 100 starts, or the engine overhauled after 500 hours.

IntelliJect provides an interface for indicating that maintenance has been done, clearing the error bit. At the same time an electronic logbook entry is made indicating the service, when it was done, and who did it. Maintenance workers can also add their own free form entries to the electronic logbook.



Engine Wear for #1234

Time	Revs	Peak CHT	Hot time	Load time	Starts
123.00 hr	10.46 M	156 C	2.10 hr	20.00 hr	56

Serial # 1234

	Status	Serviced	At	By	Description
1)	OK	2024/3/29 18:05:09	123.00 hours run time	Bob	Replace spark plug at 50 hrs or when gap > 0.028" (0.71mm), when run time exceeds 50.00 hours.
2)	OK	2024/3/29 18:05:22	123.00 hours run time	Bob	Replace fuel filter, when run time exceeds 100.00 hours.
3)	OK	2024/3/29 18:05:27	123.00 hours run time	Bob	Wash/Clean air filter and renew filter oil, when run time exceeds 100.00 hours.
4)	OK				inspect piston for scuffing if CHT exceeds 220degC for any time, when CHT exceeds 220 C.
5)	D				inspect piston if hot time exceeded, when hot time exceeds 0.10 hours.

Maintenance is triggered by hot time