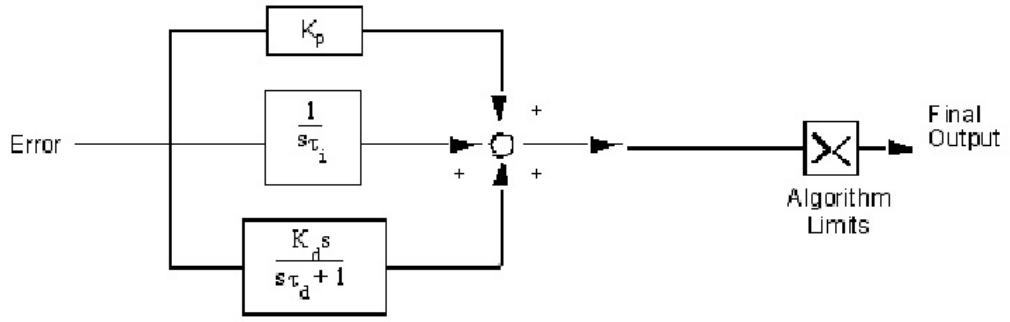
Ovation的PID算法INHB功能的分析

Zhu Wei

2015年7月6日　v1.0

1　前述

首先我们来详细讨论一下Ovation的PID的计算过程，如下图方式的并行计算：



where:

K_p = Proportional gain (PGAIN)
τ_i = Reset time (INTG)
K_d = Derivative gain (DGAIN)
τ_d = Derivative rate time constant (DRATE)
s = LaPlace operator

Note:
Output is limited by algorithm limits.

这里我们做一些简化，通常PID的微分项是不用的，将PID变为PI。假设PI是Indirect型，即偏差Error=设定值SP-过程值PV。比例增益PGAIN(K_p)为1，积分时间INTG(τ_i)为10秒。这样PI的输出OUT=$(K_p * Error) + \frac{1}{\tau_i}\int Error\, d(t)$。通常PID在下游的MASTATION算法为手动时，跟踪MASTATION的输出，此时设定值跟踪过程值，PID的偏差为0；在MASTATION切换至自动时，PID输出保持原值。如果此时偏差值发生变化，PID开始计算，实际上PID这时的输出值是原输出值加上当前偏差值计算出的输出值。Ovation的PID由控制器计算，必然受控制周期影响，假设当前逻辑运行在100毫秒的控制区。对于每个周期来说PID的输

出=投入自动时的初始输出+偏差计算输出。由于积分时间为10秒，每周期的积分量=1/10个周期/10秒*偏差，每秒的积分量=1/10秒*偏差。这里假设设定值SP和过程值PV都等于50，PID输出也为50（这里所有值都假设归一化为0~100，Ovation要求PID的SP和PV都归一化至0~100），PID下游的MASTATION刚投入自动，看下面的表格，展示PID如何计算：

时间轴（秒）	设定值SP	过程值PV	PID输出值
0	50	50	50
1	50	40	61
2	50	40	62
...	...	...	...
10	50	40	70
11	50	50	70

然而实际的Ovation控制器计算过程会稍有偏差，第一个周期的输出实际上=投入自动时的初始输出+偏差计算输出（第一个周期积分只计算半个周期），也就是说偏差改变后第1个控制周期的输出是60.05，由于Ovation广播数据是1秒一次，时间趋势上看到的值也可能是60.15,60.25,60.35,...60.95这样的值，当然如果逻辑是运算在1000ms的控制区，实际的PID输出值便是60.5。下面的图便是一个运行在100ms控制区的PID的运算结果，实际的PID输出第1秒是60.25，也就是计算了3个周期（其中第一个周期的积分项只算一半）后广播数据，当然为了简化，假设了PID输出变化，PV也没有发生变化。

Graph | **Table View** | Radar View | Information

Date Time	SP.UNIT...	PV.UNIT...	PID-OUT....
07/14/2015 09:24:31	50.000	40.000	75.250
07/14/2015 09:24:30	50.000	40.000	74.250
07/14/2015 09:24:29	50.000	40.000	73.250
07/14/2015 09:24:28	50.000	40.000	72.250
07/14/2015 09:24:27	50.000	40.000	71.250
07/14/2015 09:24:26	50.000	40.000	70.250
07/14/2015 09:24:25	50.000	40.000	69.250
07/14/2015 09:24:24	50.000	40.000	68.250
07/14/2015 09:24:23	50.000	40.000	67.250
07/14/2015 09:24:22	50.000	40.000	66.250
07/14/2015 09:24:21	50.000	40.000	65.250
07/14/2015 09:24:20	50.000	40.000	64.250
07/14/2015 09:24:19	50.000	40.000	63.250
07/14/2015 09:24:18	50.000	40.000	62.250
07/14/2015 09:24:17	50.000	40.000	61.250
07/14/2015 09:24:16	50.000	40.000	60.250
07/14/2015 09:24:15	50.000	50.000	50.000

接下来我们继续那个运行在100ms控制区的实际PID，如果在运行一段时间后PV值发生了变化呢，比如下面左边的图PV从40变回50，导致偏差为0，输出从72.95变为63，减少了9.95，这是由于比例部分由原来的1*10变为0*10，减少了10，而积分还会计算半个周期，就是10-0.05,结果是输出减少了9.95；再比如下面右边的图PV从40先变到45，再变到50。看下面的图就会发现，PV由40变为45后，PID输出由71.45变为67.275，减少了4.175，这是由于偏差由10减少了到5，这样输出在1秒内会减少5（注意之前设置的比例系数为1），而这时积分作用还在，1秒积分作用0.5，半个周期为0.025。最后PV由45变为50，PID输出最后停在64.05，相对于上个周期68.775减少了4.725。由于广播时间的原因，偏差减少了5，输出值的减少就会在4和5之间，半个周期的积分作用导致了输出会有0.025的部分。

Graph | Table View | Radar View | Information

Date Time	SP.UNIT...	PV.UNIT...	PID-OUT....
07/14/2015 09:58:35	50.000	50.000	63.000
07/14/2015 09:58:34	50.000	50.000	63.000
07/14/2015 09:58:33	50.000	50.000	63.000
07/14/2015 09:58:32	50.000	50.000	63.000
07/14/2015 09:58:31	50.000	50.000	63.000
07/14/2015 09:58:30	50.000	40.000	72.950
07/14/2015 09:58:29	50.000	40.000	71.950
07/14/2015 09:58:28	50.000	40.000	70.950
07/14/2015 09:58:27	50.000	40.000	69.950
07/14/2015 09:58:26	50.000	40.000	68.950
07/14/2015 09:58:25	50.000	40.000	67.950
07/14/2015 09:58:24	50.000	40.000	66.950
07/14/2015 09:58:23	50.000	40.000	65.950
07/14/2015 09:58:22	50.000	40.000	64.950
07/14/2015 09:58:21	50.000	40.000	63.950
07/14/2015 09:58:20	50.000	40.000	62.950
07/14/2015 09:58:19	50.000	40.000	61.950
07/14/2015 09:58:18	50.000	40.000	60.950
07/14/2015 09:58:17	50.000	50.000	50.000
07/14/2015 09:58:16	50.000	50.000	50.000

Date Time	SP.UNIT...	PV.UNIT...	PID-OUT....
07/14/2015 09:47:51	50.000	50.000	64.050
07/14/2015 09:47:50	50.000	50.000	64.050
07/14/2015 09:47:49	50.000	50.000	64.050
07/14/2015 09:47:48	50.000	50.000	64.050
07/14/2015 09:47:47	50.000	45.000	68.775
07/14/2015 09:47:46	50.000	45.000	68.275
07/14/2015 09:47:45	50.000	45.000	67.775
07/14/2015 09:47:44	50.000	45.000	67.275
07/14/2015 09:47:43	50.000	40.000	71.450
07/14/2015 09:47:42	50.000	40.000	70.450
07/14/2015 09:47:41	50.000	40.000	69.450
07/14/2015 09:47:40	50.000	40.000	68.450
07/14/2015 09:47:39	50.000	40.000	67.450
07/14/2015 09:47:38	50.000	40.000	66.450
07/14/2015 09:47:37	50.000	40.000	65.450
07/14/2015 09:47:36	50.000	40.000	64.450
07/14/2015 09:47:35	50.000	40.000	63.450
07/14/2015 09:47:34	50.000	40.000	62.450
07/14/2015 09:47:33	50.000	40.000	61.450
07/14/2015 09:47:32	50.000	40.000	60.450
07/14/2015 09:47:31	50.000	50.000	50.000

2 Hard Inhibit功能

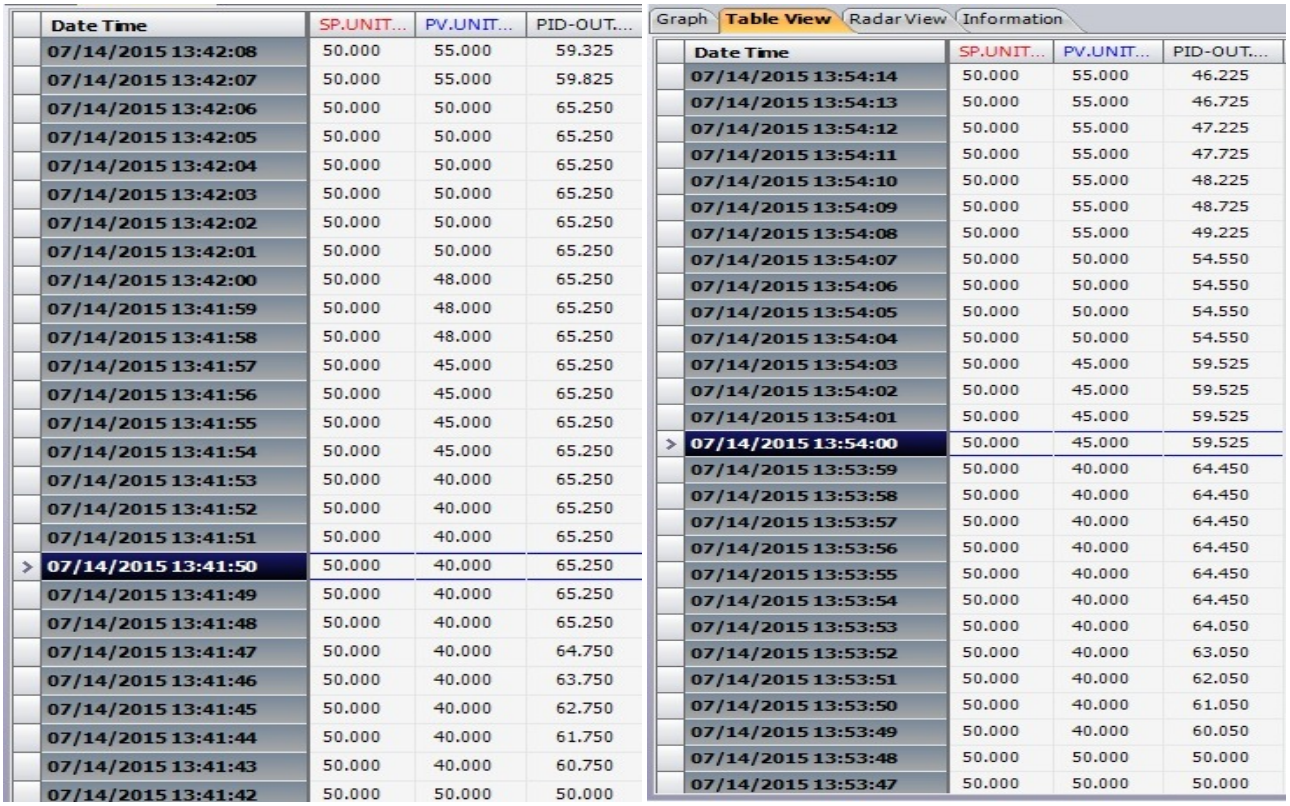
Ovation在Solaris版本的1.8开始，为PID引入了INHB参数，即Hard Inhibit参数，可以设置为Enable或Disable。Ovation使用MAMODE可以发出RAI（禁止升）和LWI（禁止降）信号给MASTATION算法，PID的跟踪输入从MASTATION的跟踪输出点接收了RAI和LWI信号。当RAI发生时，PID输出不能再增大，只能减少；当LWI发生时，PID输出不能减少，只能增

大。而Hard Inhibit功能就是在PID收到RAI或LWI信号时，行为可以不同。如果将INHB设置为Enable，在发生RAI和LWI时，PID会停止计算，输出保持最后一次的计算值，直到偏差值反向（就是符号改变，比如偏差由1变为-1或-1变为1）或禁止信号撤销。如果将INHB设置为Disable，在发生RAI和LWI时，PID还会继续计算新的输出值，如果新计算的输出值违反了禁止条件，输出会保持，如果新的值没有违反禁止条件，PID的输出值更新为新的计算值。不管INHB功能是否开启，PID的抗积分饱和功能一直作用，也就是说，输出值在遇到禁止条件发生时，会停止积分计算。

在Ovation的1.8版本之前PID没有INHB参数设置，而实际上PID是按照INHB为Enable来计算的，而PIDFF则一直是按照INHB为Disable来计算的，即使到Ovation3.5版本，PIDFF也是没有INHB参数设置功能的。

3 Hard Inhibit功能的实际表现

首先我们来看看PID的INHB功能开启(Enable)的情况，还是原来那个100ms控制器的PID恢复到初始状态投入自动，如下面左边的图，PV先到40，然后等待PID的输出到65.25的时候，发出RAI信号，这时候PID的输出就保持在65.25了，然后改变PV值从40->45->48->50，PID的输出一直不变，一直到PV变成55了，这时偏差改变符号，输出才由65.25变为59.825。这里我们就可以看到，在INHB开启时遇到RAI，然后PID输出被锁住，一直到偏差符号改变才重新计算。再让PID恢复到初始状态，但是将INHB功能关闭(Disable)，如下面右边的图，PV先到40，然后等待PID输出到64.45，发出RAI信号，这时候输出也会保持在64.45，而实际上PID内部还在计算输出值，但是因为已经到达上限，由于PID的抗积分饱和功能，积分项已经不再运算了，如果此时改变PV到45，注意这种改变并没有让偏差反向，但是PID的输出由64.45变为59.525，减少了4.925，实际上就是比例计算的PID输出由原来的1*10变为1*5，减少了5，再加上积分项的计算（一个半周期就是0.075），然后PV再由45变为50，输出由59.525减少到54.55，输出减少了4.975，实际上就是比例计算的PID输出由原来的1*5变为1*0，减少了5，再加上积分项的计算（半个周期就是0.025），积分项计算多少个周期在偏差没有变为0时是不固定的，取决于指令发生的时间，当然最多也就一秒加半个周期，而偏差变为0的那刻，积分项必然只计算半个周期。当然接下来PV变为55，偏差符号改变，这时候输出也必然再次减少，减少的值必然至少5.025。



Date Time	SP.UNIT...	PV.UNIT...	PID-OUT....
07/14/2015 13:42:08	50.000	55.000	59.325
07/14/2015 13:42:07	50.000	55.000	59.825
07/14/2015 13:42:06	50.000	50.000	65.250
07/14/2015 13:42:05	50.000	50.000	65.250
07/14/2015 13:42:04	50.000	50.000	65.250
07/14/2015 13:42:03	50.000	50.000	65.250
07/14/2015 13:42:02	50.000	50.000	65.250
07/14/2015 13:42:01	50.000	50.000	65.250
07/14/2015 13:42:00	50.000	48.000	65.250
07/14/2015 13:41:59	50.000	48.000	65.250
07/14/2015 13:41:58	50.000	48.000	65.250
07/14/2015 13:41:57	50.000	45.000	65.250
07/14/2015 13:41:56	50.000	45.000	65.250
07/14/2015 13:41:55	50.000	45.000	65.250
07/14/2015 13:41:54	50.000	45.000	65.250
07/14/2015 13:41:53	50.000	40.000	65.250
07/14/2015 13:41:52	50.000	40.000	65.250
07/14/2015 13:41:51	50.000	40.000	65.250
07/14/2015 13:41:50	50.000	40.000	65.250
07/14/2015 13:41:49	50.000	40.000	65.250
07/14/2015 13:41:48	50.000	40.000	65.250
07/14/2015 13:41:47	50.000	40.000	64.750
07/14/2015 13:41:46	50.000	40.000	63.750
07/14/2015 13:41:45	50.000	40.000	62.750
07/14/2015 13:41:44	50.000	40.000	61.750
07/14/2015 13:41:43	50.000	40.000	60.750
07/14/2015 13:41:42	50.000	50.000	50.000

Graph | Table View | Radar View | Information

Date Time	SP.UNIT...	PV.UNIT...	PID-OUT....
07/14/2015 13:54:14	50.000	55.000	46.225
07/14/2015 13:54:13	50.000	55.000	46.725
07/14/2015 13:54:12	50.000	55.000	47.225
07/14/2015 13:54:11	50.000	55.000	47.725
07/14/2015 13:54:10	50.000	55.000	48.225
07/14/2015 13:54:09	50.000	55.000	48.725
07/14/2015 13:54:08	50.000	55.000	49.225
07/14/2015 13:54:07	50.000	50.000	54.550
07/14/2015 13:54:06	50.000	50.000	54.550
07/14/2015 13:54:05	50.000	50.000	54.550
07/14/2015 13:54:04	50.000	50.000	54.550
07/14/2015 13:54:03	50.000	45.000	59.525
07/14/2015 13:54:02	50.000	45.000	59.525
07/14/2015 13:54:01	50.000	45.000	59.525
07/14/2015 13:54:00	50.000	45.000	59.525
07/14/2015 13:53:59	50.000	40.000	64.450
07/14/2015 13:53:58	50.000	40.000	64.450
07/14/2015 13:53:57	50.000	40.000	64.450
07/14/2015 13:53:56	50.000	40.000	64.450
07/14/2015 13:53:55	50.000	40.000	64.450
07/14/2015 13:53:54	50.000	40.000	64.450
07/14/2015 13:53:53	50.000	40.000	64.050
07/14/2015 13:53:52	50.000	40.000	63.050
07/14/2015 13:53:51	50.000	40.000	62.050
07/14/2015 13:53:50	50.000	40.000	61.050
07/14/2015 13:53:49	50.000	40.000	60.050
07/14/2015 13:53:48	50.000	50.000	50.000
07/14/2015 13:53:47	50.000	50.000	50.000

4 实际使用中遇到的问题

上面说了那么多，分析了Ovation的PID实际计算过程，接下来我们就说说实际使用中的问题。常常有用户会抱怨PIDFF的问题，由于PIDFF的默认行为就是INHB为关闭的行为，而且PIDFF也没有设置INHB为开启的功能。试想一个使用PIDFF(INDIRECT类型)的场景，输出值在50的时候，RAI信号发出，而此时过程值PV一直小于设定值，我们认为PIDFF的输出就应该保持在RAI信号发出的那一刻，然而由于此时PID还在计算，即使过程值PV小于设定值，但是一直在波动，那么不断重新计算的比例项，只要遇到偏差减少的情况，就会减少输出值（即使此时偏差的符号还没有改变），而且由于抗积分饱和的功能，RAI信号在的时候，积分项几乎不被计算，最后输出值就会达到下限（如果RAI信号一直在的话）。

那可以说PIDFF这是一种bug吗，当然不是。对于PIDFF来说，FF的开环前馈计算是起主要作用的，PID的闭环计算只是一个修正作用，用于微调。另外RAI或者LWI信号一直发着也是有问题的，说明你的前馈计算有问题，稳不住过程，而一个好的控制策略，应该说，前馈的计算值是多少，最终的输出就是多少，PID的闭环修正只是偶尔作用一下，更不要说还要

一直来个RAI或者LWI了，说明被控对象根本稳不住吗。那么PID原来默认INHB开启的情况不是好好的么，为什么还要引入关闭功能呢，必然是某些情况下的需求咯，比如过程的滞后较大，如果被RAI或LWI锁住后，要一直等到偏差反向才动作，就有可能晚了。

上面一段说了那么多，其实没什么用，如果被部分工艺党看到了，一定会说搞系统的啥啥都不知道，然后就变成口水战了。

5 总结

如果有人对PIDFF的行为感到困惑，那就让他用PID配SUM吧，PID默认INHB为Enable。如果有人对那半个周期的积分项感到困惑，这其实也很好理解，想想微积分里面$\int_0^1 x\,dx = \frac{1}{2} + C$，这里设常数C为0。

. . . 没有啦，就到这里吧。