NAME

at_pid − proportional/integral/derivative controller with auto tuning

SYNOPSIS

loadrt at_pid [num_chan=*num* | **names=***name1***[,***name2...***]]**

DESCRIPTION

at_pid は、古典的な比例/積分/微分コントローラーであり、サーボモーターやその他の閉ループアプリケーションの位置または速度のフィードバックループを制御するために使用されます。

at_pid は、最大 16 個のコントローラーをサポートします。 実際にロードされる数は、モジュールがロードされるときに num_chan 引数によって設定されます。 または、names =と一意の名前をコンマで区切って指定します。

num_chan =および names =指定子は相互に排他的です。 num_chan =も names =も指定されていない場合、デフォルト値は 3 です。

debug が 1 に設定されている場合（デフォルトは 0）、いくつかの追加の HAL パラメーターがエクスポートされます。これはチューニングに役立つ場合がありますが、それ以外の場合は不要です。

at_pid には自動調整モードが組み込まれています。 これは、プロセスを特徴付けるリミットサイクルを設定することによって機能します。 このことから、Pgain / Igain / Dgain または Pgain / Igain / FF1 は、Ziegler-Nichols を使用して決定できます。 FF1 を使用する場合は、出力がユーザー単位/秒になるようにスケーリングを設定する必要があります。

オートチューニング中は、コマンド入力は変更しないでください。 リミットサイクルは、指令された位置の周りに設定されます。 自動チューニングを開始するために初期チューニング値は必要ありません。 オートチューニングを開始する前に設定する必要があるのは、**tune−cycles**、**tune−effort**、tune-mode のみです。 オートチューニングが完了すると、チューニングパラメータが設定されます。 LinuxCNC から実行している場合、次のエラーを回避するために、調整対象の軸の FERROR 設定をリミットサイクル振幅よりも大きくする必要があるため、緩める必要がある場合があります。

オートチューニングを行うには、以下の手順で行ってください。 調整する軸を、移動の中心近くのどこかに移動します。 tune-cycles（ほとんどの場合デフォルト値で問題ありません）と tune-mode を設定します。 tune-effort を小さい値に設定します。 enable を true に設定します。 tune-mode を true に設定します。 tune-start を true に設定します。 振動が発生しない場合、または振動が小さすぎる場合は、tune-effort をゆっくりと増やします。 オートチューニングは、enable または tune-mode を false に設定することでいつでも中止できます。

NAMING

ピン、パラメーター、および関数の名前には、次の接頭辞が付いています。

pid.N. for N=0,1,...,num−1 when using **num_chan=num**

nameN. for nameN=name1,name2,... when using **names=name1,name2,...**

pid.N. 以下の説明にフォーマットを示します。

FUNCTIONS

pid.*N*.do−pid−calcs (uses floating-point)

Does the PID calculations for control loop *N*.

PINS

pid.*N*.command float in

制御ループに必要な（コマンドされた）値。

pid.*N*.feedback float in

エンコーダーなどのセンサーからの実際の（フィードバック）値。

pid.*N*.error float out

コマンドとフィードバックの差。

pid.*N*.output float out

モーターなどのアクチュエーターに送られる PID ループの出力。

pid.*N*.enable bit in

true の場合、PID 計算を有効にします。 false の場合、出力はゼロになり、すべての内部積分器などがリセットされます。

pid.*N*.tune−mode bit in

true の場合、自動調整モードを有効にします。 false の場合、通常の PID 計算が実行されます。

pid.*N*.tune−start bit io

true に設定すると、自動チューニングを開始します。 オートチューニングが完了するとクリアされます。

PARAMETERS

pid.*N*.Pgain float in

比例ゲイン。 結果は、エラーに Pgain を掛けたものである出力への寄与になります。

pid.*N*.Igain float in

積分ゲイン。 結果は、エラーに Igain を掛けた積分である出力への寄与になります。 たとえば、10 秒間続いた 0.02 のエラーは 0.2 の積分エラー（errorI）になり、Igain が 20 の場合、積分項は出力に 4.0 を追加します。

pid.*N*.Dgain float in

微分ゲイン。 結果は、エラーの変化率（導関数）に Dgain を掛けたものである出力への寄与になります。 たとえば、0.2 秒で 0.02 から 0.03 に変化したエラーは、0.05 のエラー導関数（errorD）になり、Dgain が 5 の場合、導関数項は出力に 0.25 を追加します。

pid.*N*.bias float in

バイアスは、出力に追加される一定量です。 ほとんどの場合、ゼロのままにしておく必要があります。 ただし、サーボアンプのオフセットを補正したり、垂直方向に移動するオブジェクトの重量のバランスをとったりすると便利な場合があります。 出力の他のすべてのコンポーネントと同様に、PID ループが無効になると、バイアスはオフになります。 PID ループが無効になっている場合でもゼロ以外の出力が必要な場合は、外部 HALsum2 ブロックを追加する必要があります。

pid.*N*.FF0 float in

ゼロ次フィードフォワード項。 FF0 にコマンド値を掛けたものが出力に寄与します。 位置ループの場合、通常はゼロのままにしておく必要があります。 速度ループの場合、FF0 は摩擦またはモーターカウンター EMF を補正でき、適切に使用すればより良いチューニングが可能になる場合があります。

pid.*N*.FF1 float in

一次フィードフォワード項。 FF1 にコマンド値の導関数を掛けた出力への寄与を生成します。 位置ループの場合、寄与は速度に比例し、摩擦またはモーター CEMF を補正するために使用できます。 速度ループの場合、加速度に比例し、慣性を補正できます。 どちらの場合も、適切に使用すると、チューニングが向上する可能性があります。

pid.*N*.FF2 float in

2 次フィードフォワード項。 FF2 にコマンド値の 2 次導関数を掛けた出力への寄与を生成します。 位置ループの場合、寄与は加速度に比例し、慣性を補正するために使用できます。 速度ループの場合、寄与はジャークに比例し、通常はゼロのままにする必要があります。

pid.*N*.deadband float in

「許容可能な」エラーの範囲を定義します。 エラーの絶対値が不感帯よりも小さい場合、エラーがゼロであるかのように扱われます。 本質的に量子化されたエンコーダーなどのフィードバックデバイスを使用する場合、コマンドが 2 つの隣接するエンコーダー値の間にある場合に制御ループが前後にハンチングするのを防ぐために、不感帯は半分のカウントよりわずかに大きく設定する必要があります。 エラーの絶対値が不感帯よりも大きい場合、不感帯のエッジでの伝達関数のステップを防ぐために、ループ計算を実行する前に不感帯の値がエラーから差し引かれます。（バグを参照してください。）

pid.*N*.maxoutput float in

出力制限。 maxoutput がゼロでない限り、出力の絶対値が maxoutput を超えることはできません。 出力が制限されている場合、ワインドアップとオーバーシュートを防ぐために、エラー積分器は積分する代わりに保持します。

pid.*N*.maxerror float in

P、I、および D に使用される内部エラー変数の制限。エラーが大きい場合（コマンドがステップ変更を行う場合など）に、高い Pgain 値が大きな出力を生成しないようにするために使用できます。 通常は必要ありませんが、非線形システムを調整するときに役立ちます。

pid.*N*.maxerrorD float rw

エラー導関数の制限。 Dgain 項で使用されるエラーの変化率は、値がゼロでない限り、この値に制限されます。 Dgain の影響を制限し、コマンドやフィードバックのステップによる大きな出力スパイクを防ぐために使用できます。 通常は必要ありません。

pid.*N*.maxerrorI float rw

エラー積分器の制限。 Igain 項で使用されるエラー積分器は、ゼロでない限り、この値に制限されます。 インテグレータのワインドアップと、エラーの発生中または発生後に発生するオーバーシュートを防ぐために使用できます。 通常は必要ありません。

pid.*N*.maxcmdD float rw

コマンド派生物の制限。 FF1 で使用されるコマンド導関数は、値がゼロでない限り、この値に制限されます。 コマンドにステップ変更がある場合に、FF1 が大きな出力スパイクを生成するのを防ぐために使用できます。 通常は必要ありません。

pid.*N*.maxcmdDD float rw

コマンドの二次導関数の制限。 FF2 で使用されるコマンドの 2 次導関数は、値がゼロでない限り、この値に制限されます。 コマンドにステップ変更がある場合に、FF2 が大きな出力スパイクを生成するのを防ぐために使用できます。 通常は必要ありません。

pid.*N*.tune−type u32 rw

0 に設定すると、Pgain / Igain / Dgain が計算されます。 1 に設定すると、Pgain / Igain / FF1 が計算されます。

pid.*N*.tune−cycles u32 rw

プロセスを特徴づけるために実行するサイクル数を決定します。 tune-cycles は、実際には半サイクルの数を設定します。 すべてのサイクルの平均が使用されるため、サイクルが多いほど、より正確な特性評価が得られます。

pid.*N*.tune−effort float rw

プロセスでリミットサイクルを設定する際に使用される効果を決定します。 tune-effort は、maxoutput よりも小さい正の値に設定する必要があります。 小さなものから始めて、最大モーター電流のかなりの部分が使用される結果になる値まで作業します。 値が小さいほど、リミットサイクルの振幅は小さくなります。

pid.*N*.errorI float ro (only if debug=1)

エラーの積分。 これは、出力の積分項を生成するために Igain を掛けた値です。

pid.*N*.errorD float ro (only if debug=1)

エラーの派生物。 これは、出力の微分項を生成するために Dgain を掛けた値です。

pid.*N*.commandD float ro (only if debug=1)

コマンドの派生物。 これは、出力の 1 次フィードフォワード項を生成するために FF1 を掛けた値です。

pid.*N*.commandDD float ro (only if debug=1)

コマンドの 2 次導関数。 これは、出力の 2 次フィードフォワード項を生成するために FF2 を掛けた値です。

pid.*N*.ultimate−gain float ro (only if debug=1)

プロセスの特性から決定されます。 アルティメットゲインは、リミットサイクル振幅に対するチューンエフォートの比率に 4.0 を掛けて Pi で割ったものです。 pid.N.ultimate-period float ro（debug = 1 の場合のみ）プロセスの特性から決定されます。 Ultimate-period は、リミットサイクルの期間です。

BUGS

一部の人々は、デッドバンドは、エラーがデッドバンド内にある場合はゼロとして扱われ、デッドバンド外にある場合は変更されないように実装する必要があると主張します。 これは、不感帯のサイズに等しい伝達関数のステップを引き起こすため、実行されませんでした。 その振る舞いを好む人は、振る舞いを変更するパラメーターを追加するか、独自のバージョンの at_pid を作成することを歓迎します。 ただし、デフォルトの動作は変更しないでください。