ARIS-EC

ロボティクス向け分散型リアルタイム・エラー補正アーキテクチャ

Absolute Reduction Integration Sequence – Error Correction

ロボットのセンシング、制御、アクチュエーションに対する階層型ランタイム補正

教育および技術的オリエンテーションを目的として作成

Althea Project

2026 年 5 月

配布に関する注記

本書は、一般教育および技術的オリエンテーションを目的として作成されています。保護対象の方程式、導出、制限された実装アーキテクチャ、または機密ランタイム・ロジックを開示することなく、ロボット・システムにおける ARIS-EC の実装概念を説明します。

要旨

現代のロボット・システムは、センサー入力、知覚、予測、コマンド発行、アクチュエーター実行、フィードバック確認という連続したチェーンに依存している。このチェーンの各段階でエラーが発生し得る。エラーには、センサーノイズ、タイミング・ドリフト、キャリブレーション不一致、アクチュエーター遅延、コマンドと実行の偏差、フィードバック不安定性など、ロボット内部で生じるものがある。また、電磁干渉、無線周波数ノイズ、熱、湿気、振動、粉塵、圧力変動、照明の不安定性、その他の外乱など、動作環境から加わるエラーもある。

ARIS-EC は、Absolute Reduction Integration Sequence – Error Correction のランタイム・プロファイルであり、ロボットの入出力経路全体に分散型リアルタイム補正アーキテクチャとして実装できる。公開用のエンジニアリング表現では、ARIS-EC は、ロボットのデータ・ループおよび実行ループ内の測定可能なエラー状態に作用する、制約付きで監査可能な補正層と説明するのが最も適切である。ARIS-EC は、ロボットのセンサー、制御スタック、安全コントローラー、ドライブ電子回路、アクチュエーターを置き換えるものではない。それらを接続する情報チェーンとコマンド・チェーンの信頼性を高める。

本書は、ロボット・システムにおける ARIS-EC の公開用実装モデルを提示する。主要な補正層を定義し、基本的な数学的関係を導入し、センサー分解能、アクチュエーター応答、コントローラー帯域幅、機械的精度の向上に伴って ARIS-EC が自然に拡張できることを説明する。本モデルは意図的に公開安全型として構成されており、一般的な補正アーキテクチャを超える保護対象の内部ランタイム詳細は開示しない。

1. はじめに

ロボットは現実から直接行動するのではない。解釈されたデータに基づいて行動する。

ロボットはセンサーから入力を受け取り、その入力を知覚へ変換し、近未来のシステム状態を予測し、コマンドを発行し、アクチュエーターによって動作を実行し、フィードバックによって結果を確認する。これにより、次の連続ループが形成される。

センサー入力 → 知覚 → 予測 → コマンド → アクチュエーション → フィードバック

実世界の動作環境では、このループの各段階でエラーが累積し得る。小さなセンサー偏差が知覚エラーになり、知覚エラーが予測エラーになり、予測エラーがコマンド・エラーになり、コマンド・エラーがアクチュエーター実行エラーになる。これらの偏差が十分迅速に補正されなければ、ロボットは非効率に動作し、応答が遅れ、障害物を誤分類し、過剰補正または補正不足を起こし、あるいは危険な状態に入る可能性がある。

ARIS-EC は、この問題に制約付きリアルタイム補正プロファイルとして対処する。その機能は、ロボット・システム内部の対応可能な偏差を特定し、その偏差を運用上許容できるゼロ状態へ近づけることである。ロボティクスへの導入では、ARIS-EC はランタイム・スパインに適用される補正プロファイルとして機能し、入力、ルーティング、エスカレーション、出力、監査、エクスポート動作に制約を設定する。

その結果、公開安全型のロボティクス・アーキテクチャが成立する。ARIS-EC は、ホスト・ロボットの物理的限界を超えない範囲で、センサーデータ、環境外乱、知覚ー制御ロジック、アクチュエーター実行、フィードバック確認にわたるエラーを補正し、ロボットの信頼性を向上させることができる。

2. 中核実装原理

ARIS-EC ロボティクスの中心原理は、次のとおりである。

期待状態 – 実状態 = 対応可能エラー

あらゆるロボット・サブシステムについて、ARIS-EC は、そのサブシステムの期待状態と、測定・推定・フィードバック確認された実状態を比較する。その差分が補正対象のエラー項となる。

一般形は、次のとおりである。

$$E(t) = X_{expected}(t) - X_{actual}(t)$$

ここで、

$E(t)$	=	時刻 t におけるリアルタイム・エラー
$X_{expected}(t)$	=	時刻 t における意図された、または予測されたシステム状態
$X_{actual}(t)$	=	時刻 t における測定・推定・フィードバック確認されたシステム状態

公開用の補正目標は、次のとおりである。

$$E(t) \to OZ$$

ここで、OZ = Operational Zero（オペレーショナル・ゼロ）である。Operational Zero とは、対応可能エラーが、ロボットの運用目的、安全しきい値、ハードウェア限界、測定精度、および用途要件にとって問題となる水準を下回るまで低減された状態を意味する。すべての物理ノイズが消失したという意味ではない。残存偏差が、ロボットにとって意味のある補正境界を下回ったことを意味する。

この区別は重要である。ARIS-EC は、物理ハードウェアの限界を超えることを主張しない。ホスト・プラットフォームの測定可能な限界内で、補正の整合性を向上させる。

3. 階層型ロボティクス・アーキテクチャとしての ARIS-EC

ARIS-EC のロボティクス実装は、次の 4 つの主要補正層に編成できる。

- 環境干渉層
- センサー経路層
- 知覚ー制御層
- 駆動／アクチュエーター層

これらの層は一体となり、ロボットの入出力チェーン全体に分散型補正アーキテクチャを形成する。

全経路は、次のように表せる。

環境 → センサー・ハードウェア → ARIS-EC_environmental → ARIS-EC_sensor → ARIS-EC_perception-control → ARIS-EC_actuator-drive → 検証済みロボット動作

この構造により、ARIS-EC は、エラーが上位の意思決定または物理実行を汚染する前に、そのエラーを低減できる。

ランタイムでは、この階層モデルは、次の制約付き実装パターンに明確に対応する。

Layer 0:	受信したロボット状態を、使用可能な補正形式へ正規化する
Layer I:	支配的なエラー・シグネチャを検出し、分類する
Layer II:	許可された補正経路を実行する
Layer III:	補正された状態を検証する
Layer IV:	補正済みモデル状態をロボットの物理挙動と同期させる
Layer V:	制約付き MAP 出力、監査データ、または補正指示を出力する

公開用の説明では、これを決定論的な「検出ー補正ー検証ー同期ー出力」ループと表現できる。

4. Layer 1：環境干渉補正

ロボットは、電気的、熱的、機械的、光学的に中立ではない環境で動作する。環境外乱は、ロボットがデータを解釈する前に、そのデータを歪める可能性がある。

一般的な環境外乱の分類には、次が含まれる。

EMF、RF ノイズ、熱、湿気、振動、粉塵、圧力変動、照明変動

これらの外乱は、センサー、配線、通信バス、プロセッサ、電源システム、アクチュエーター・フィードバック・チャネルに影響を及ぼし得る。

環境エラー項は、次のように表せる。

$$E_{env}(t) = D_{expected}(t) - D_{disturbed}(t)$$

ここで、

$E_{env}(t)$	=	環境によって誘発されたデータ・エラー
$D_{expected}(t)$	=	想定されるクリーンなデータ・ストリーム
$D_{disturbed}(t)$	=	環境干渉の影響を受けたデータ・ストリーム

補正目標は、次のとおりである。

$$E_{env}(t) \to OZ$$

補正された環境経路は、次のようになる。

$$D_{raw}(t) \to D_{environment\text{-}corrected}(t)$$

この層は、産業用ロボット、屋外ロボット、自動運転車、ドローン、警備ロボット、倉庫ロボット、医療ロボット、および不安定またはノイズの多い環境で動作するその他のシステムにおいて、特に重要である。

5. Layer 2：センサー経路補正

環境干渉が低減された後、次の補正対象はセンサー経路そのものである。

センサーは、それ自体のエラー分類を生じさせる。これには、キャリブレーション・ドリフト、タイミング不一致、サンプリング・エラー、レイテンシ、分解能限界、偽陽性、偽陰性、信号破損、センサー・フュージョンの不一致、パケット・レベルのデータ不整合などが含まれ得る。

センサー・エラー項は、次のように表せる。

$$E_{sensor}(t) = S_{expected}(t) - S_{actual}(t)$$

ここで、

$E_{sensor}(t)$	=	センサー経路エラー
$S_{expected}(t)$	=	期待されるセンサー出力
$S_{actual}(t)$	=	実際のセンサー出力

補正目標は、次のとおりである。

$$E_{sensor}(t) \to OZ$$

センサー経路は、次のように表せる。

$$S_{raw}(t) \to S_{corrected}(t)$$

簡略化した補正式は、次のとおりである。

$$S_{corrected}(t) = S_{raw}(t) - E_{sensor}(t)$$

実装上、この層は、ロボットの知覚システムへ入るデータの品質を向上させる。よりクリーンなデータは、知覚、予測、計画、制御の各層にかかる負担を軽減する。

6. Layer 3：知覚－制御補正

知覚－制御層は、中心的な行動補正層である。ここで ARIS-EC は、ロボットによる現実の解釈、近未来予測、コマンド生成が、意図された安全運用状態と整合しているかを評価する。

この層は、補正済みデータからロボットの意思決定への遷移を扱う。

$$S_{corrected}(t) \rightarrow \text{知覚} \rightarrow \text{予測} \rightarrow \text{コマンド}$$

一般的な運動制御エラー項は、次のように表せる。

$$E_{motion}(t) = M_{expected}(t) - M_{actual}(t)$$

ここで、

$E_{motion}(t)$	=	運動状態エラー
$M_{expected}(t)$	=	期待される安全な運動状態
$M_{actual}(t)$	=	実際の、または予測された運動状態

補正目標は、次のとおりである。

$$E_{motion}(t) \rightarrow OZ$$

この層は、障害物回避、動的経路補正、緊急制動、速度調整、バランス補正、ルート調整、物体操作、衝突防止、人近接安全に適用できる。

動的障害物への応答では、簡略化した関係を次のように表せる。

$$E_{safe}(t) = P_{safe}(t) - P_{projected}(t)$$

ここで、

$P_{safe}(t)$	=	安全な予測位置または状態
$P_{projected}(t)$	=	現在のコマンドに基づくロボットの予測位置または状態

したがって、

$$E_{safe}(t) \rightarrow OZ$$

平易なエンジニアリング表現では、ARIS-EC は、ロボットが安全上行うべきことと、実際に行っていること、または行おうとしていることとの差を継続的に低減する。

7. Layer 4：駆動およびアクチュエーター補正

ロボットの故障は、ロボットが誤って知覚することだけによって起こるのではない。誤って実行することによっても起こり得る。

コマンドが正しくても、モーターが遅れることがある。ブレーキの応答が遅すぎることがある。サーボがオーバーシュートすることがある。関節に負荷変動が生じることがある。車輪が滑ることがある。ドローンのローター性能が不足することがある。グリッパーが期待された力を加えられないことがある。油圧または空圧サブシステムは、温度、圧力、摩耗条件によって異なる応答を示すことがある。

アクチュエーター・エラー項は、次のように表せる。

$$E_{actuator}(t) = A_{commanded}(t) - A_{executed}(t)$$

ここで、

$E_{actuator}(t)$	=	アクチュエーター実行エラー
$A_{commanded}(t)$	=	指令されたアクチュエーター挙動
$A_{executed}(t)$	=	測定されたアクチュエーター挙動

補正目標は、次のとおりである。

$$E_{actuator}(t) \to OZ$$

アクチュエーター経路は、次のようになる。

$$A_{commanded}(t) \to A_{verified}(t)$$

ここで、

$A_{verified}(t)$ = フィードバックにより確認されたアクチュエーター実行

安全なロボット挙動は、正しい意思決定だけでなく、検証済みの物理実行にも依存するため、この層は不可欠である。

8. 統合 ARIS-EC ロボット・ランタイム・モデル

4 つの補正層を統合すると、ロボット補正スタックは次のようになる。

$$D_{raw}(t) \to D_{environment\text{-}corrected}(t) \to S_{corrected}(t) \to C_{corrected}(t) \to A_{verified}(t)$$

ここで、

$D_{raw}(t)$	=	受信した生データ・ストリーム
$D_{environment\text{-}corrected}(t)$	=	環境干渉に対して補正されたデータ
$S_{corrected}(t)$	=	センサー経路補正済みデータ
$C_{corrected}(t)$	=	補正されたコマンドまたは制御状態
$A_{verified}(t)$	=	フィードバック確認済みのアクチュエーター出力

ロボットの総エラーは、階層和として次のようにモデル化できる。

$$E_{total}(t) = E_{env}(t) + E_{sensor}(t) + E_{motion}(t) + E_{actuator}(t)$$

システム・レベルの補正目標は、次のとおりである。

$$E_{total}(t) \to OZ$$

この公開用表現は意図的に簡略化されている。実際のシステムでは、これらのエラー項は結合し、非線形、断続的、または文脈依存となる場合がある。たとえば、環境ノイズがセンサー・エラーを生み、センサー・エラーが知覚エラーを生み、アクチュエーター遅延が次の知覚サイクルへフィードバックされることがある。したがって、ARIS-EC は、ロボットを分離したモジュール群ではなく、接続された補正システムとして扱う。

9. 検出、ルーティング、再分類

ARIS-EC は、すべてのロボット・エラーが 1 つの固定分類に属すると仮定しない。支配的なエラー・シグネチャを検出し、補正経路をルーティングし、残差パターンが元の経路の不完全さを示した場合には再分類する。

公開用の表現では、ランタイムは次の残差を評価する。

$$R_n(t) = Psi_n(t) - Psi_{n-1}(t)$$

ここで、

$R_n(t)$	=	現在の補正サイクル後に残るエラー
$Psi_n(t)$	=	現在の補正済みシステム状態
$Psi_{n-1}(t)$	=	前のシステム状態

簡略化した残差減衰比は、次のとおりである。

$$Delta\ R_n = ||R_n|| / ||R_{n-1}||$$

残差が急速に減衰する場合、現在の補正経路を継続する。残差が停滞、振動、移動、別軸での反発、調和パターンの形成、または複数サブシステムへの出現を示す場合、ARIS-EC は、EC 許可プロファイルに従ってエラー経路を再分類し、補正をルーティングできる。

ロボット・システムでは、これは次を意味する。

振動	→	時間補正経路
空間的移動	→	ネストまたは再帰補正経路
軸方向の反発	→	直交回転補正経路
調和包絡	→	調和補正経路
サブシステム横断残差	→	連結システム補正経路

これはロボティクスにおいて特に重要である。単一の観測故障でも、複数の原因が考えられるためである。たとえば車輪スリップの問題は、アクチュエーター問題のように見えても、実際には床面状態センシング、タイミング・ドリフト、環境干渉に起因する場合がある。ARIS-EC の補正価値の一部は、すべてのエラーを最初に見える原因から生じたものとして扱うことを避ける能力にある。

10. 検証と物理同期

ロボット補正層は、補正が実際に機能したことを検証できなければ有用ではない。

したがって ARIS-EC は、補正後に検証段階を必要とする。公開用のエンジニアリング表現では、補正済みモデル状態が、許容された運用公差内で、ロボットの測定された物理挙動と一致しなければならないことを意味する。

検証関係は、次のように表せる。

$$|Y_{predicted}(t) - Y_{measured}(t)| <= epsilon_{operational}$$

ここで、

$Y_{predicted}(t)$	=	期待される補正済み出力
$Y_{measured}(t)$	=	センサーで確認された物理出力
$epsilon_{operational}$	=	用途固有の許容公差

この検証プロセスの後に同期が行われる。同期とは、ロボット内部の補正済み状態と、物理ロボットの実際の挙動が整合したままであることをランタイムが確認することを意味する。数理的な補正が完了しているように見えても、物理システムには遅延、機械的反発、アクチュエーター応答の遅れ、環境外乱が残っている場合があるため、稼働中のロボット・システムではこの点が重要である。

> ***補正状態と物理挙動が公差内で一致するまで、補正は完了していない。***

11. 監査可能性と公開安全型制約プロファイル

ARIS-EC は、無制約の完全 ARIS 導入ではなく、制約付きランタイム・プロファイルとして公開説明されるべきである。

これはロボティクスにおいて重要である。安全システムには、決定論的挙動、明確な監査記録、制限された出力、予測可能な統合境界が必要だからである。したがって ARIS-EC は、次を統制する公開安全型制約プロファイルの下で動作する。

入力 → ルーティング → エスカレーション → 出力 → エクスポート

実務上、ARIS-EC は次を保持しなければならない。

TraceID、入力参照、ルート履歴、残差履歴、検証結果、出力判断

これにより、エンジニアは、エラー検出から補正結果までの監査可能な経路を得られる。

公開用コミュニケーションで重要なのは、ARIS-EC が単なるポリシー文言ではなく、実装によって制約されるよう設計されている点である。ロボティクス向け出力は、許可された補正推奨、制御調整、フィルタリングされた証拠要約、監査フック、およびホスト・システムに適した MAP 形式の規定出力に限定されるべきである。

12. ハードウェア制約による精度境界

ARIS-EC は、ロボットの物理的限界を取り除かない。実用上の最大補正精度は、ロボットのハードウェアと制御インフラストラクチャによって制約される。

この境界は、次のように表せる。

A_ARIS-EC <= min(A_sensors, A_actuators, A_drive latency, A_controller bandwidth, A_mechanical limit)

ここで、

A_ARIS-EC	=	実効 ARIS-EC 補正精度
A_sensors	=	センサー分解能および信頼性の限界
A_actuators	=	アクチュエーター精度の限界
A_drive latency	=	駆動システムのタイミング限界
A_controller bandwidth	=	制御ループ処理の限界
A_mechanical limit	=	物理的な機械応答の限界

この境界は、信頼できる公開用コミュニケーションに不可欠である。ARIS-EC は、補正の整合性を高め、対応可能エラーを低減し、リアルタイム応答を最適化できる。しかし、センサーの測定範囲外の情報を検出させることはできず、アクチュエーターに物理応答能力を超えさせることもできない。

ただし、ロボット・ハードウェアが向上するにつれて、ARIS-EC の補正範囲も自然に拡大する。

13. センサーおよびアクチュエーター開発に伴う拡張

センサーの時間感度、空間精度、環境ノイズ耐性が高まるほど、ARIS-EC はより高品質な入力を受け取る。

アクチュエーターが高速化、高精度化、高動的応答化するほど、ARIS-EC はより細粒度の補正コマンドを発行し、より精密なフィードバック確認を受け取ることができる。

この拡張関係は、次のように表せる。

Delta t_sensor 低下、Delta t_actuator 低下、v_response 上昇
したがって：C_ARIS-EC 上昇

ここで、

Delta t_sensor	=	センサーのサンプリングまたは検出間隔
Delta t_actuator	=	アクチュエーター応答間隔
v_response	=	ロボット応答速度
C_ARIS-EC	=	実効リアルタイム補正能力

これは、ARIS-EC が単一のロボティクス・プラットフォームまたは固定世代のハードウェアに依存しないことを意味する。ARIS-EC は、ホスト・マシンの測定可能な入出力能力に応じて拡張する補正アーキテクチャである。

14. 実用的な実装分野

ARIS-EC は、産業用ロボット、自律移動ロボット、ヒューマノイド・ロボット、ドローン、警備ロボット、倉庫ロボット、農業ロボット、医療ロボット、検査ロボット、建設ロボット、自動運転車など、複数のロボット分類に適用できる。

実装上の重点は、システムごとに異なる。

産業用ロボットでは、アクチュエーター精度、再現性、振動補正、人近接安全が優先される場合がある。

ドローンでは、風外乱、ローター応答、GPS/IMU 補正、RF 干渉、リアルタイム安定化が優先される場合がある。

倉庫ロボットでは、障害物検出、床面状態の変化、車輪スリップ、ルート補正、センサー・フュージョンの信頼性が優先される場合がある。

警備ロボットでは、環境認識、低照度知覚、RF 干渉、移動物体分類、安全な巡回挙動が優先される場合がある。

医療ロボットでは、アクチュエーター偏差、タイミング・エラー、力の不一致、フィードバック確認不良に対して、極めて低い許容度が優先される場合がある。

同じ ARIS-EC 原理が、これらすべての分野に適用される。

> 対応可能エラーを発見 → 対応可能エラーを補正 → 補正状態を検証

15. システム・レベルの価値

ロボティクスにおける ARIS-EC の主な価値は、ロボットの信頼性を単一モジュールの問題ではなく、システム全体の補正問題として扱う点にある。

センサーノイズだけ、知覚エラーだけ、経路計画だけ、またはアクチュエーター挙動だけを補正するのではなく、ARIS-EC はロボットの完全なループ全体で動作できる。

> 入力完全性 → 意思決定完全性 → 実行完全性 → フィードバック完全性

これにより、4 つの実用的利益が生じる。

第一に、ロボットはよりクリーンなデータを受け取る。

第二に、ロボットは、より正確なリアルタイム情報に基づいて判断する。

第三に、ロボットのコマンドは、意図された安全運用状態により密接に整合する。

第四に、アクチュエーター実行は、実際の物理性能に照らして検証・補正される。

その結果、安全性、応答性、精度、環境堅牢性が向上したロボット・システムが得られる。

16. 公開安全型の技術的位置づけ

ARIS-EC は、公開用に次のように位置づけることができる。

> *ARIS-EC は、ロボティクス向けの制約付き・監査可能なリアルタイム補正アーキテクチャである。環境干渉、センサーデータ経路、知覚ー制御ループ、駆動／アクチュエーター実行の全域で動作できる。その目的は、ロボットの入出力チェーン全体にわたって対応可能エラーを低減し、ホスト・プラットフォームの測定可能限界内で、よりクリーンなデータ、より正確な判断、より信頼性の高い物理実行を実現することである。*

より短い表現は、次のとおりである。

> *ARIS-EC は、センサー入力、制御判断、アクチュエーター実行、フィードバック確認の信頼性を向上させる、リアルタイムのロボット・エラー補正層である。*

さらに簡潔な表現は、次のとおりである。

> *ARIS-EC は、エラーが危険または非効率な挙動になる前に、ロボットがリアルタイムでエラーを補正することを支援する。*

17. 結論

ロボットの安全性と性能は、緊急停止だけに依存するものではない。センシング、解釈、判断、動作、フィードバックという完全なチェーン全体にわたる継続的な補正に依存する。

ARIS-EC は、ロボット・システム内部に階層型リアルタイム補正アーキテクチャとして実装できる。環境層では、EMF、RF ノイズ、熱、湿気、振動、粉塵、照明変動などの外部条件による干渉を低減する。センサー経路層では、受信データの品質を向上させる。知覚ー制御層では、期待される安全挙動と、実際または予測されたロボット挙動との偏差を低減する。駆動／アクチュエーター層では、物理実行が発行されたコマンドと一致しているかを検証する。

ARIS-EC ロボティクスの完全モデルは、次のとおりである。

$$E_{total}(t) = E_{env}(t) + E_{sensor}(t) + E_{motion}(t) + E_{actuator}(t)$$

補正目標は、次のとおりである。

$$E_{total}(t) \to OZ$$

これにより、ARIS-EC はロボティクスにおいて明確な役割を持つ。ARIS-EC は単なる緊急停止機能の強化ではない。ロボットの信頼性、安全性、精度のための分散型補正ランタイムである。

ロボットのセンサー、コントローラー、アクチュエーターが向上するにつれて、ARIS-EC も自然にそれらとともに拡張する。ロボットがより良く感知し、処理し、動作し、フィードバックを確認できるほど、ARIS-EC はリアルタイムで対応可能エラーをより効果的に低減できる。

付録 A：主要方程式

一般エラー項

$$E(t) = X_{expected}(t) - X_{actual}(t)$$

一般補正目標

$$E(t) \to OZ$$

環境エラー

$$E_{env}(t) = D_{expected}(t) - D_{disturbed}(t)$$

センサー・エラー

$$E_{sensor}(t) = S_{expected}(t) - S_{actual}(t)$$

センサー補正

$$S_{corrected}(t) = S_{raw}(t) - E_{sensor}(t)$$

運動エラー

$$E_{motion}(t) = M_{expected}(t) - M_{actual}(t)$$

安全状態エラー

$$E_{safe}(t) = P_{safe}(t) - P_{projected}(t)$$

アクチュエーター・エラー

$$E_{actuator}(t) = A_{commanded}(t) - A_{executed}(t)$$

ロボット総エラー

E_total(t) = E_env(t) + E_sensor(t) + E_motion(t) + E_actuator(t)

システム・レベル補正目標

E_total(t) → OZ

残差項

R_n(t) = Psi_n(t) − Psi_{n-1}(t)

残差減衰比

Delta R_n = ||R_n|| / ||R_{n-1}||

検証境界

|Y_predicted(t) − Y_measured(t)| <= epsilon_operational

ハードウェア制約による精度境界

A_ARIS-EC <= min(A_sensors, A_actuators, A_drive latency, A_controller bandwidth, A_mechanical limit)

拡張関係

Delta t_sensor 低下、Delta t_actuator 低下、v_response 上昇
したがって：C_ARIS-EC 上昇