Independent
Verification &
Validation

Japan Manned Space Systems Corporation
JAMSS

レジリエンス・エンジニアリングによる新しい冗長設計

有人宇宙システム株式会社
野本秀樹

Contents

- 背景
- 冗長設計の課題
- 新しい冗長設計の提案
- 制御ロジックと効率
- レジリエンス設計を導入した安全設計の特徴

背景

高信頼性システムに対する

- 高機能化
- コスト削減

要求は、年々エスカレートしている

背景

たとえば

深宇宙探査

- 想定外の多重故障に耐える高信頼性
- 圧倒的な低コスト化

を両立しなければ、そもそもミッションが成立しない

背景

しかし、高信頼性のために、冗長数を増やすと、コストは跳ね上がってゆく

両者の関係は、常にトレードオフ

背景

しかし、高信頼性のために、冗長数を増やすと、コストは跳ね上がってゆく

両者の関係は、常にトレードオフ

設計アーキテクチャをドラスティックに変えないとジレンマを脱却できない

課題

冗長設計には、他にも課題がある

課題

冗長設計には、他にも課題がある

冗長設計パラドックス

課題

冗長設計には、他にも課題がある

冗長設計パラドックス

冗長設計はハードウエアのランダム故障には有効だが、現代的で複雑な機能間インタラクション（機能共鳴）の設計ミスには効果が薄い

冗長性パラドクス
冗長設計によって安全性が低下する

"Although redundancy provides protection against accidents caused by individual component failure, it is not as effective against hazards that arise from the interactions among components in the increasingly complex and interactive systems being engineered today. ***In fact, redundancy may increase complexity to the point where the redundancy itself contributes to accidents.*** *"*

Nancy Leveson "SAFEWARE"

冗長性パラドクス
冗長設計によって安全性が低下する

NASA Dryden Flight Research Facilityが1988年に発表した新型インテリジェント航空機のフライトテスト結果：

冗長切り替えメカニズム自体の誤動作による不具合	7
冗長切り替えによって解決できなかった共通故障モード	3
原因不明	5

冗長性パラドクス
冗長設計によって安全性が低下する

NASA Dryden Flight Research Facilityが1988年に発表した新型インテリジェント航空機のフライトテスト結果：

冗長切り替えメカニズム自体の誤動作による不具合	7
冗長切り替えによって解決できなかった共通故障モード	3
原因不明	5

つまり、全ては

- 冗長設計が無ければそもそも起こらなかった不具合
- 冗長設計では解決できない不具合

課題

信頼性とコストのトレードオフから脱却し、
冗長設計パラドクスから脱却できる

そんな冗長設計アーキテクチャが必要

解決方法

解決方法

レジリエンス・エンジニアリングに基づき、自然界の冗長設計をヒントにした、冗長設計アーキテクチャを採用する

解決方法

レジリエンス・エンジニアリングに基づき、自然界の冗長設計をヒントにした、冗長設計アーキテクチャを採用する

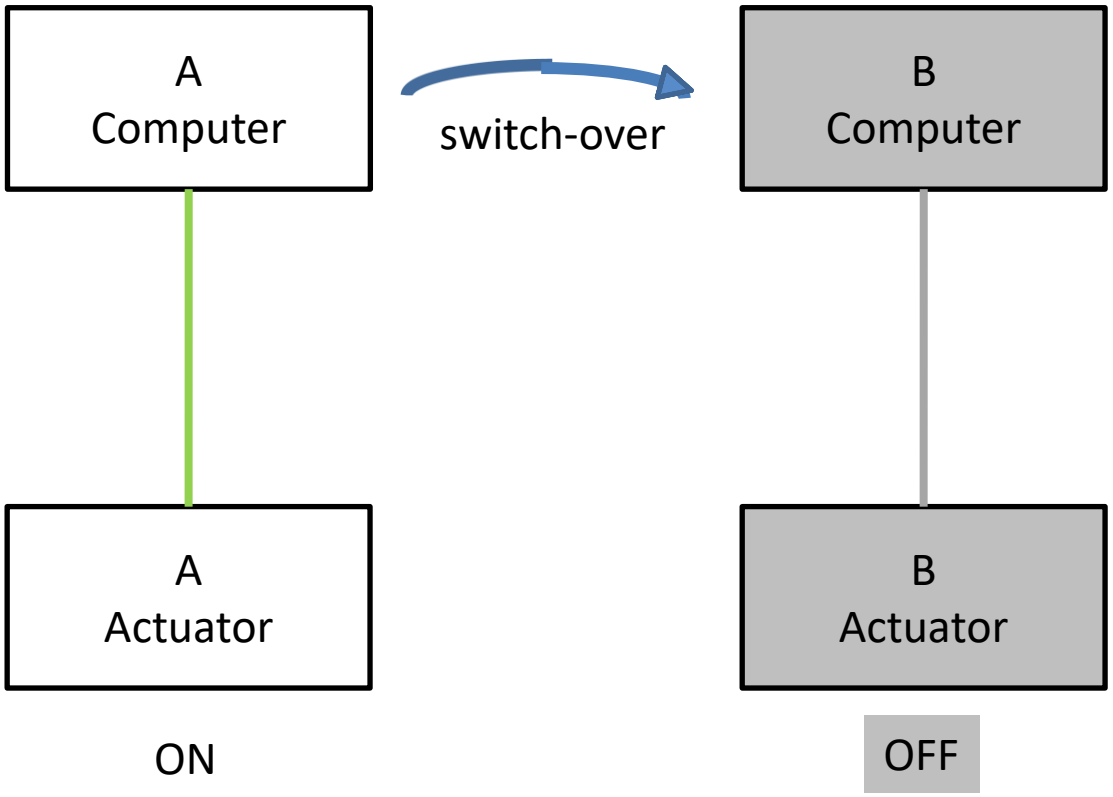
レジリエンス：

復元力、回復力、弾力

困難な状況にもかかわらず、しなやかに適応して生き延びる力

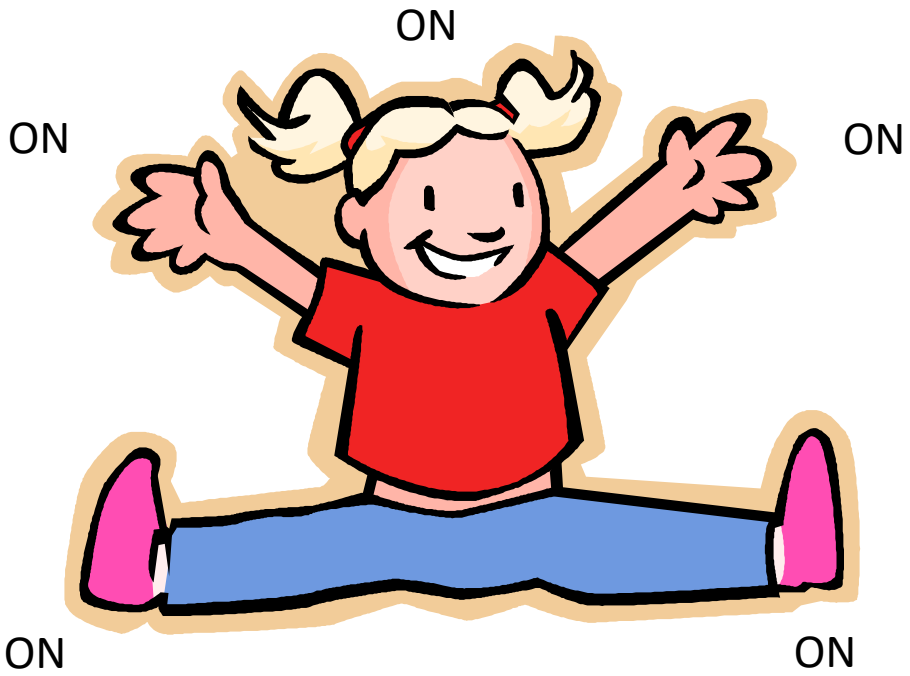
従来の冗長設計例

主系 ON.
バックアップ系 OFF.
故障でバックアップ系に切り替わる

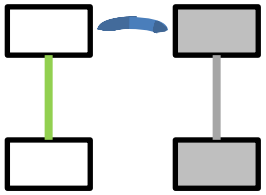


自然界の冗長設計

全てのコンポーネントがON.
バックアップ系への切り替えという概念が無い



従来型冗長設計の弱点



- 資源がフルに使えない
  - 常に半分以上が未使用資源である
- 共通故障モードに弱い
  - 主系もバックアップ系も同じ設計
  - 同じ理由で両系とも喪失する
- 冗長切り替え動作のリスクが高い
  - システムを複雑にしている最大の原因が切り替えメカニズム。特にソフトウエア制御のバグの温床

自然界の冗長設計の長所

- 全資源を常にフル活用
  - 動植物の進化の歴史は飢餓との戦い（コストとの戦い）
  - 資源を無駄にしないアーキテクチャが選ばれた
- 共通故障モードに強い
  - 右手と左手は同じ形。しかし、機能が違う
  - 機能が違えば同じ理由で喪失しにくい
- 切り替え動作特有のリスクが無い
  - オフのコンポーネントが無い
  - つまり、切り替えという動作そのものが無い

自然界の冗長設計の長所

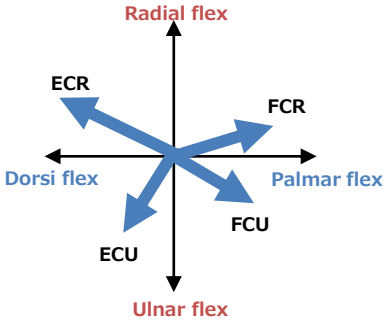
動物の**制御**

　動物の間接自由度には高い冗長性があり、一つの軸の制御には、複数の間接筋が協調する。同時に、下図のように、同一の間接筋が、複数の軸制御に係わる。

例：
ECRの動きのためには、RadialもDorsiも使われる。
Radialも、Dorsiも、ともにECR以外でも使われる。

従来、人工衛星の姿勢制御は６軸各々にユニークなアクチュエータをアサインしており、単一故障で１つの軸の制御を喪失するが、動物の制御は同一アクチュエータ数でも単一故障点が無いよう**オーバーラップ**させる。

	Radial flex	Ulnar flex	Dorsi flex	Palmar flex
ECR	○	×	○	×
ECU	×	○	○	×
FCR	△	×	×	○
FCU	×	△	×	○



東北大学医工学研究科　渡辺研究室、
"フィードバック誤差学習を用いた学習型FES制御器の開発"、
http://www.ecei.tohoku.ac.jp/fes/detail_fes2.html

新しい冗長設計の設計指針

- バックアップを無くす
- 完全同一設計を無くす
- 切り替えを無くす

バックアップを無くす

- 常にすべての機器を使う
  - 全ての機器にそれぞれの役割を持たせる
  - 常に状況を全員でシェアしているのでジャンプしない
- 切り替えでは無く、デグレード
  - 壊れたらオフにして隔離できる
  - 長期ミッションのためにオフにして温存もできる
  - オフのコンポーネント数増加に従い、緩やかにデグレードしてゆく

完全同一設計を無くす

- 共通故障モードの最小化

  機能的に異なる系を冗長として持つ

  – 例：右手と左手の冗長（ナイフとフォーク）

    - 各系に、**異なる機能**を持たせる

完全同一設計を無くす

- 共通故障モードの最小化

  機能的に異なる系を冗長として持つ

  - 例：右手と左手の冗長（ナイフとフォーク）
    - 各系に、**異なる機能**を持たせる
  - 例：脳と神経の冗長
    - 各系に、**異なる機能レイヤー**を持たせる

完全同一設計を無くす

- 共通故障モードの最小化

  機能的に異なる系を冗長として持つ

  – 例：右手と左手の冗長（ナイフとフォーク）
    - 各系に、**異なる機能**を持たせる

  – 例：脳と神経の冗長
    - 各系に、**異なる機能レイヤー**を持たせる

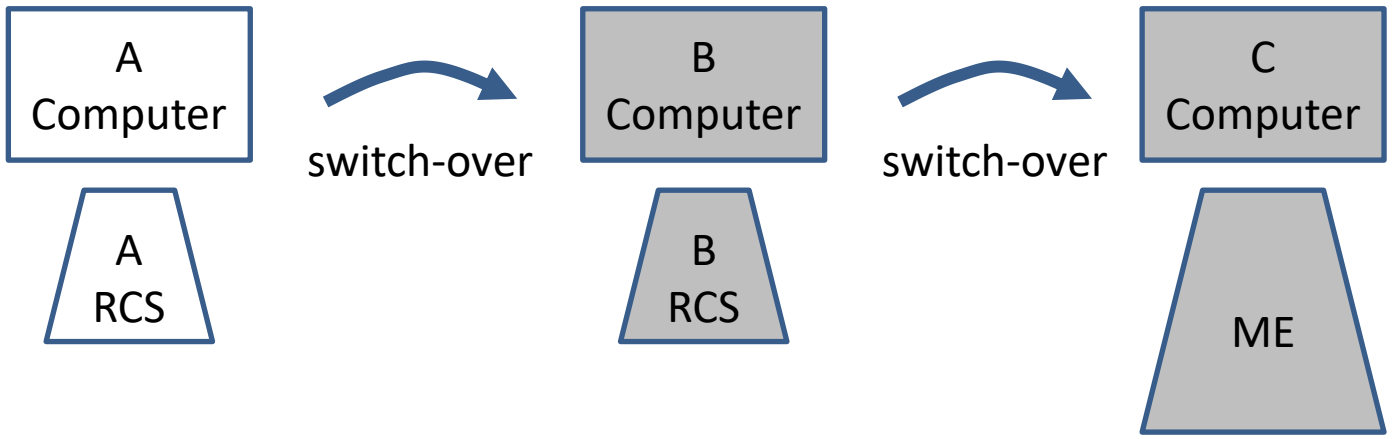
  – 例：手の指と足の指の冗長
    - 手：　掴む　書く　触る
    - 足：　掴む　　蹴る
    - 各系に、**異なる機能サブセット**を持たせる

切り替えをしない

- 冗長系設計による設計リスク:
  - 切り替え時に発生するゼロリセットや制御のジャンプ
  - 電源操作による不安定動作
- 切り替えを無くすことによるシンプルなデザイン
  - 冗長は持つが複雑性の温床となる要素を排除する
    - 冗長切り替えを排除
    - 電源のオフ･オンを排除
    - 初期化によるゼロリセット･ジャンプを排除

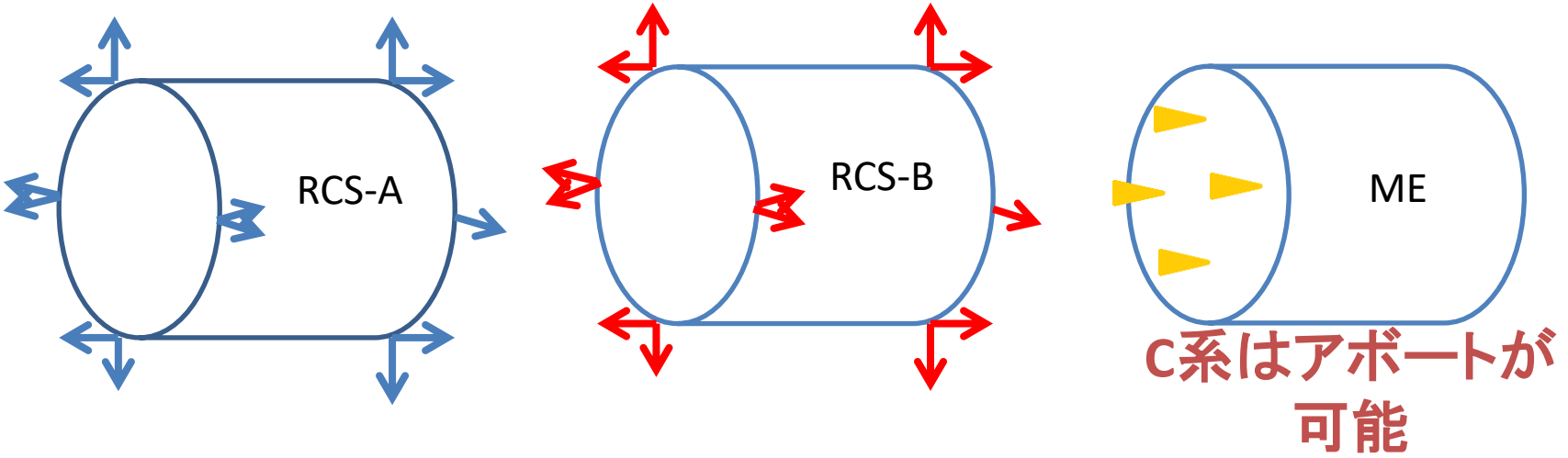
新しい冗長設計の具体化

従来型のアーキテクチャ



従来型のアーキテクチャ

A系とB系は6軸制御可能（ミッション達成可能）

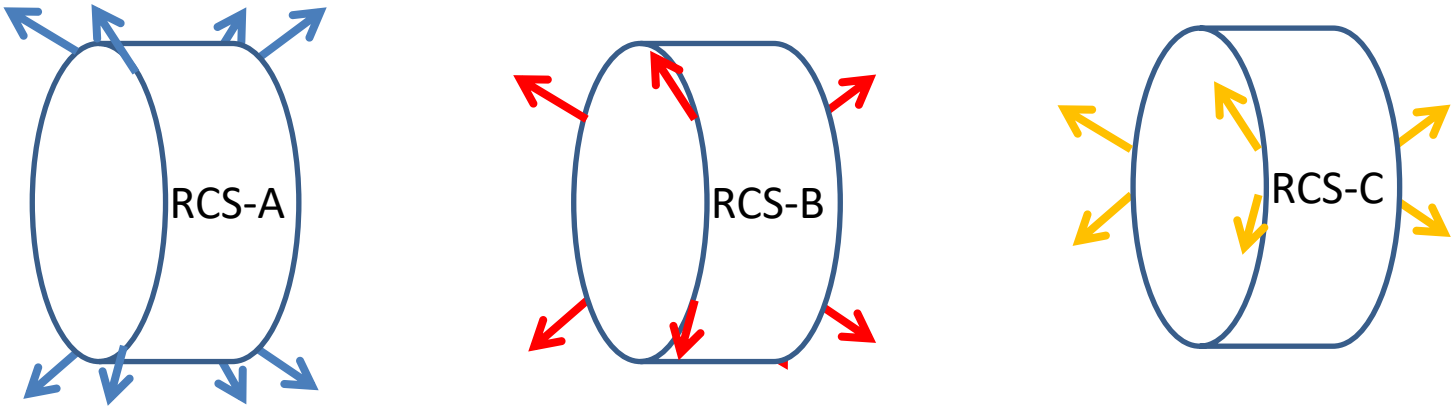


C系はアボートが可能

1 Fail Operative (2 Fail safe) by 32 スラスタ

新しいアーキテクチャ

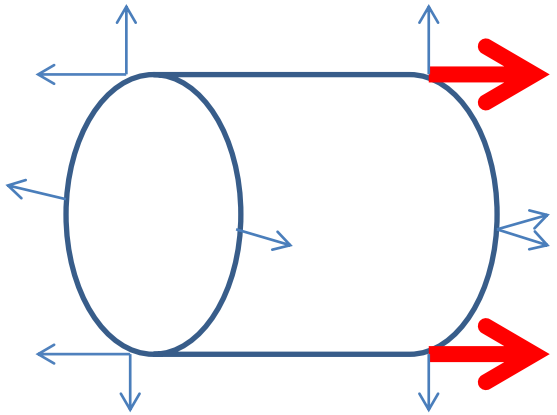
A/B/C全てが6軸制御可能（2故障でもミッション達成可能）
それぞれが、「特異な軸」を持っている（共通故障モードに強い）



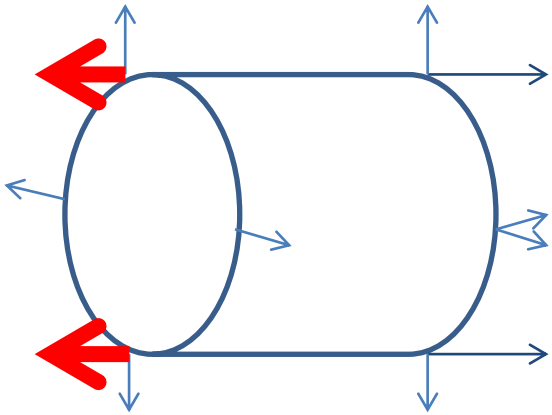
2 Fail Operative by 24 スラスター

スラスタ数25%減で、ミッション達成率100%増
＝ミッション効率150%増

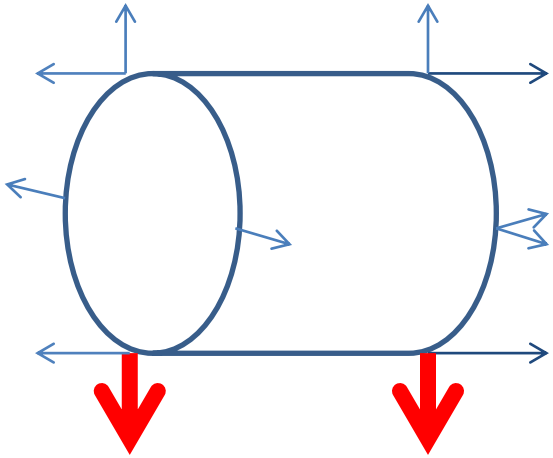
従来設計



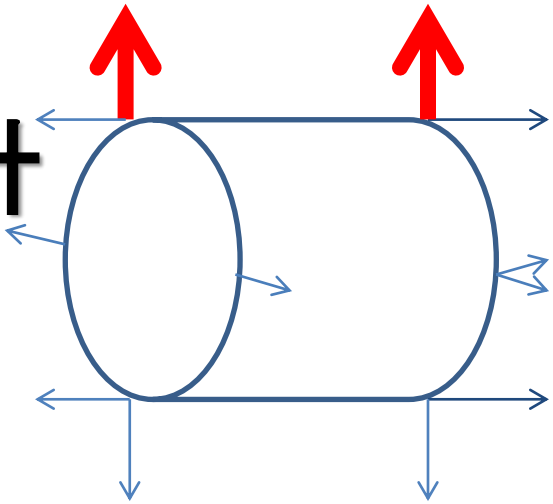
従来設計



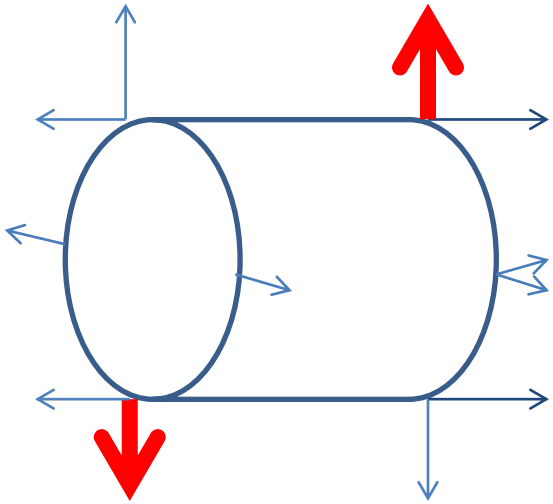
従来設計



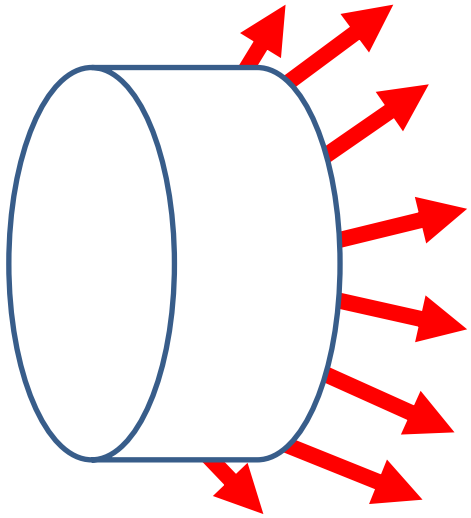
従来設計



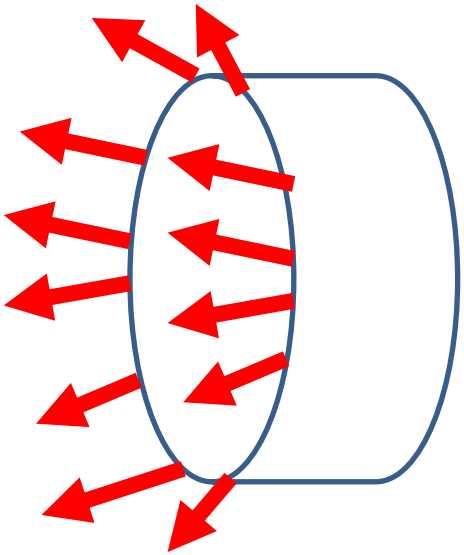
従来設計



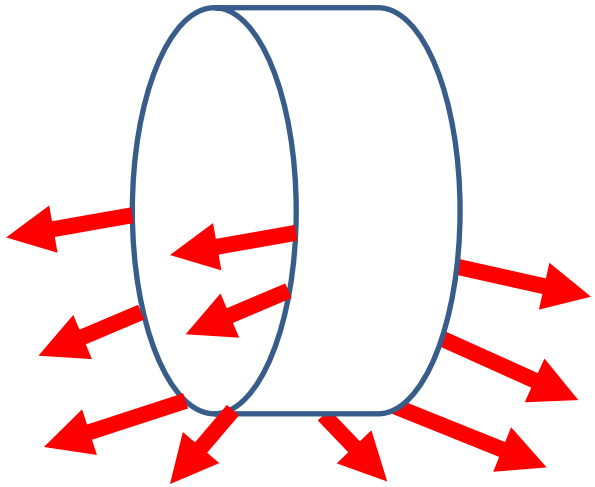
新設計



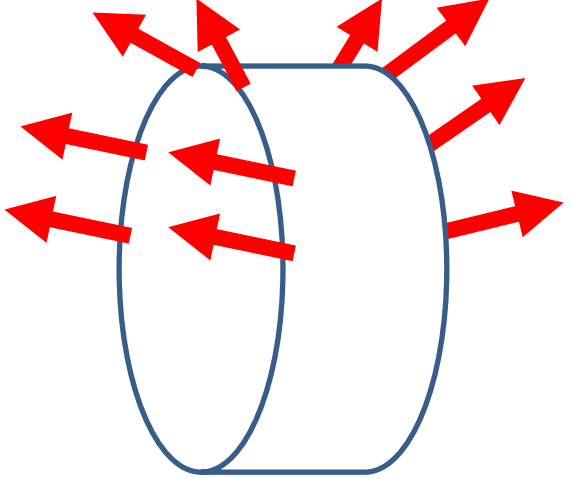
新設計



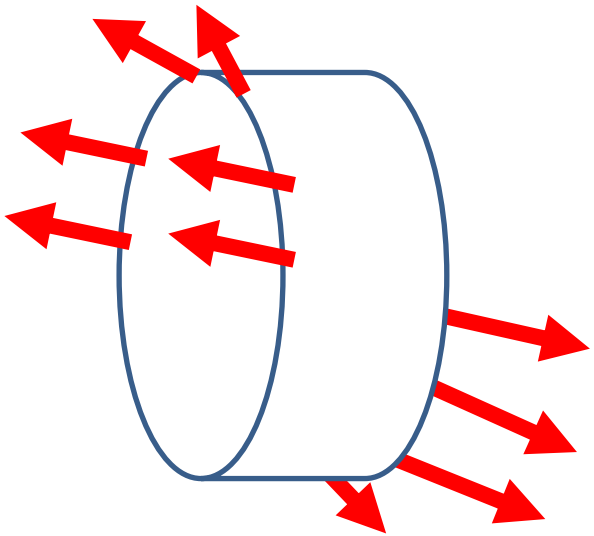
新設計



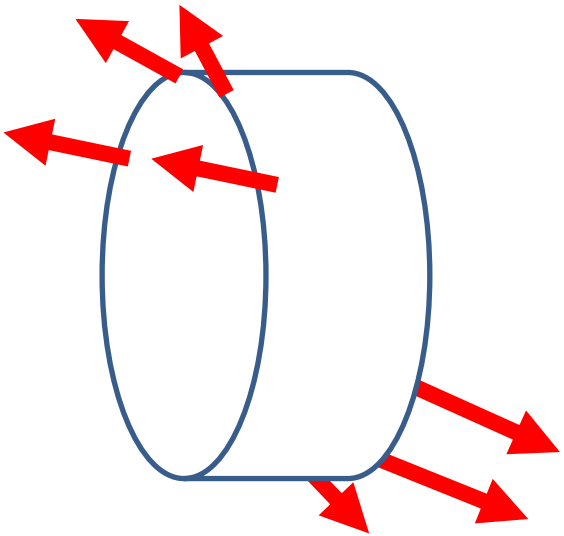
新設計



新設計

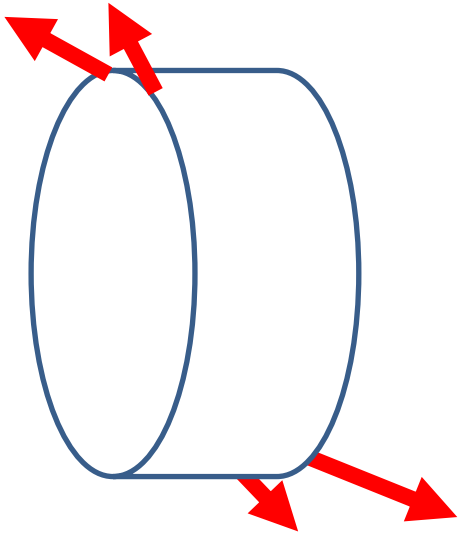


新設計



1 Failure

新設計



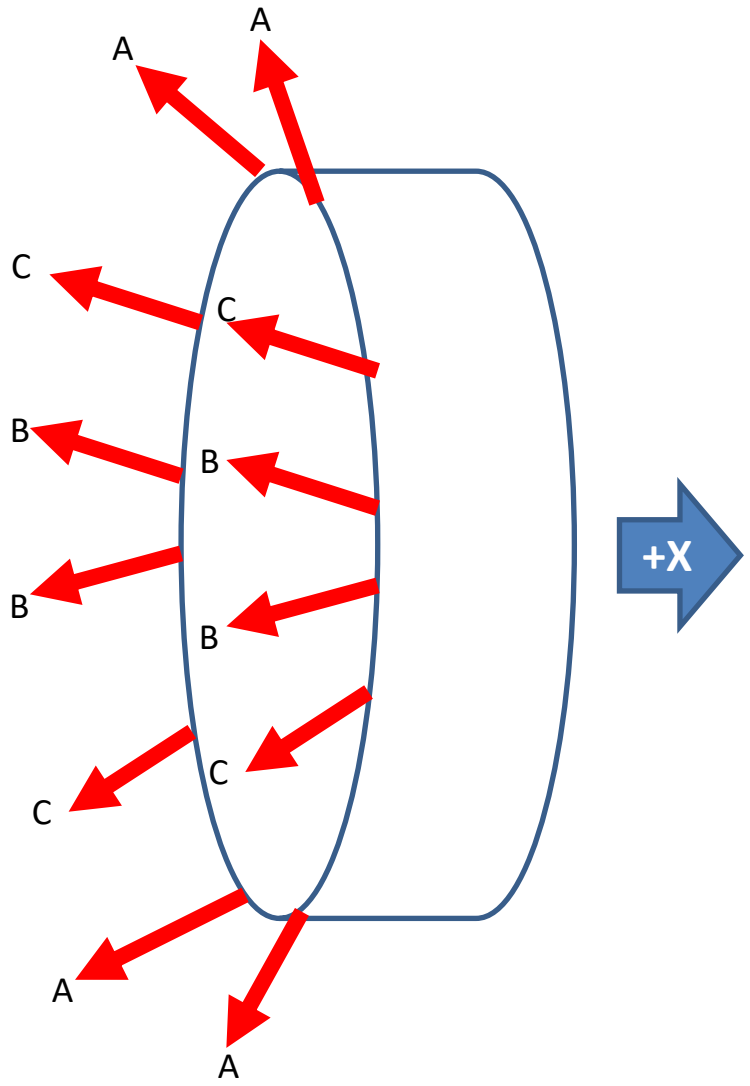
2 Failures

制御ロジック

制御ロジック

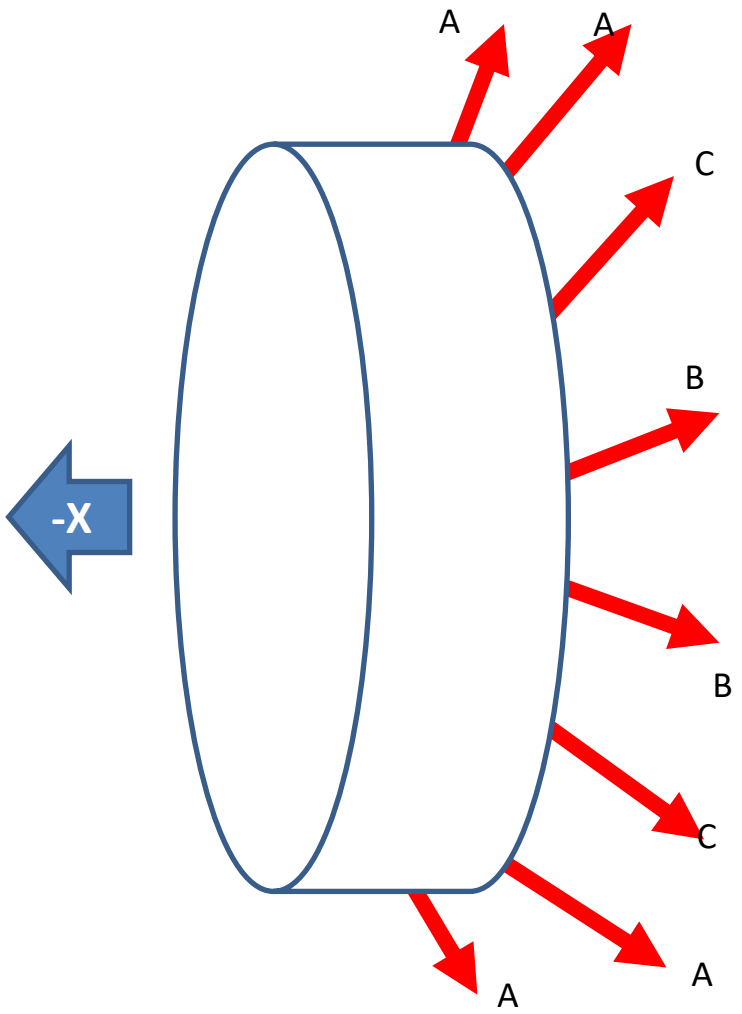
ノミナルでは、全ての系を一緒に噴射する

+X



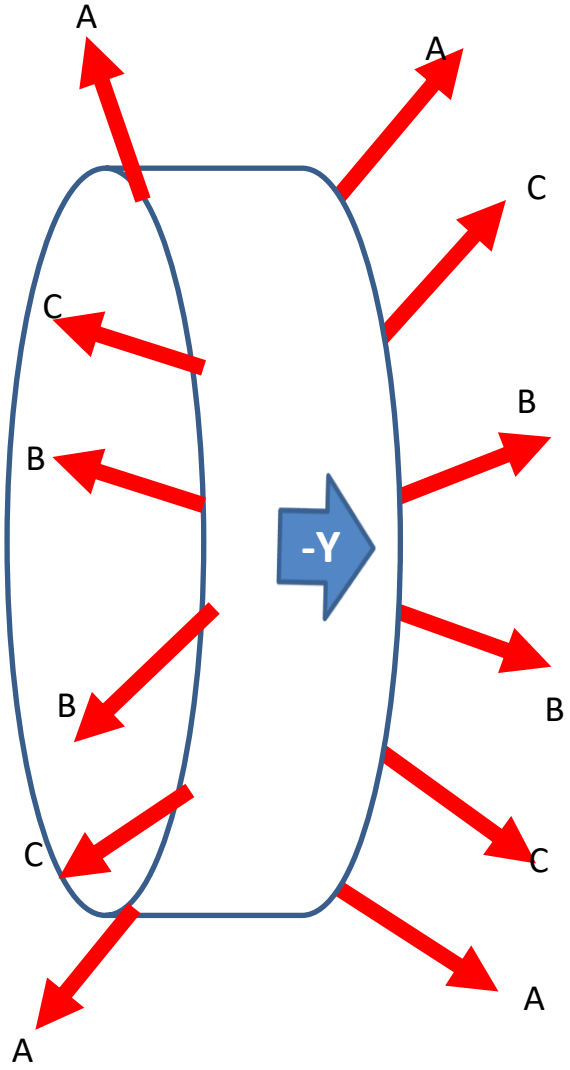
制御ロジック

-X



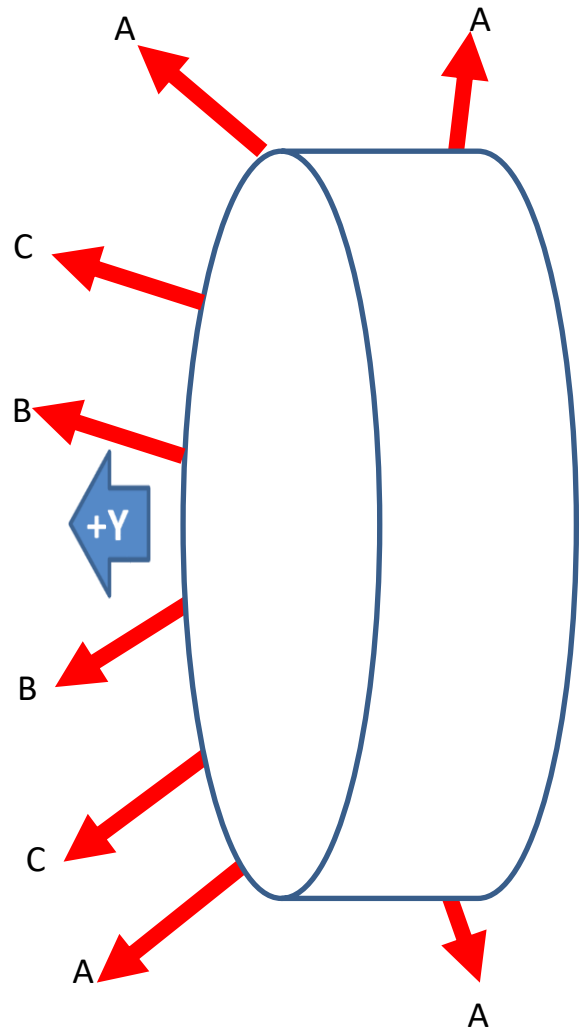
制御ロジック

-Y



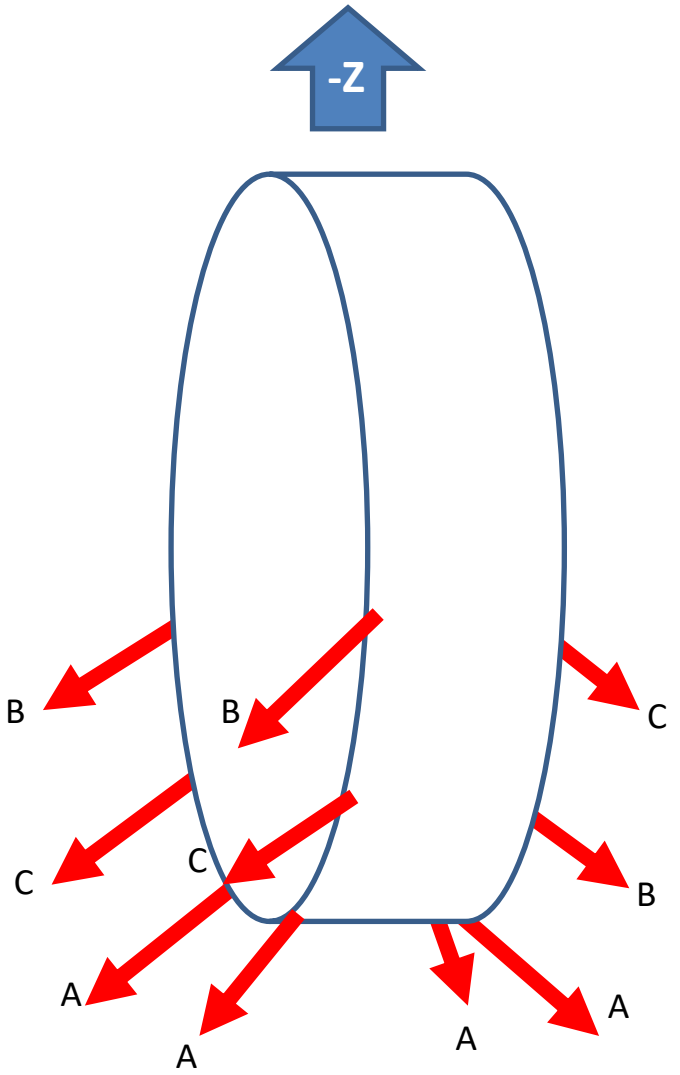
制御ロジック

+Y



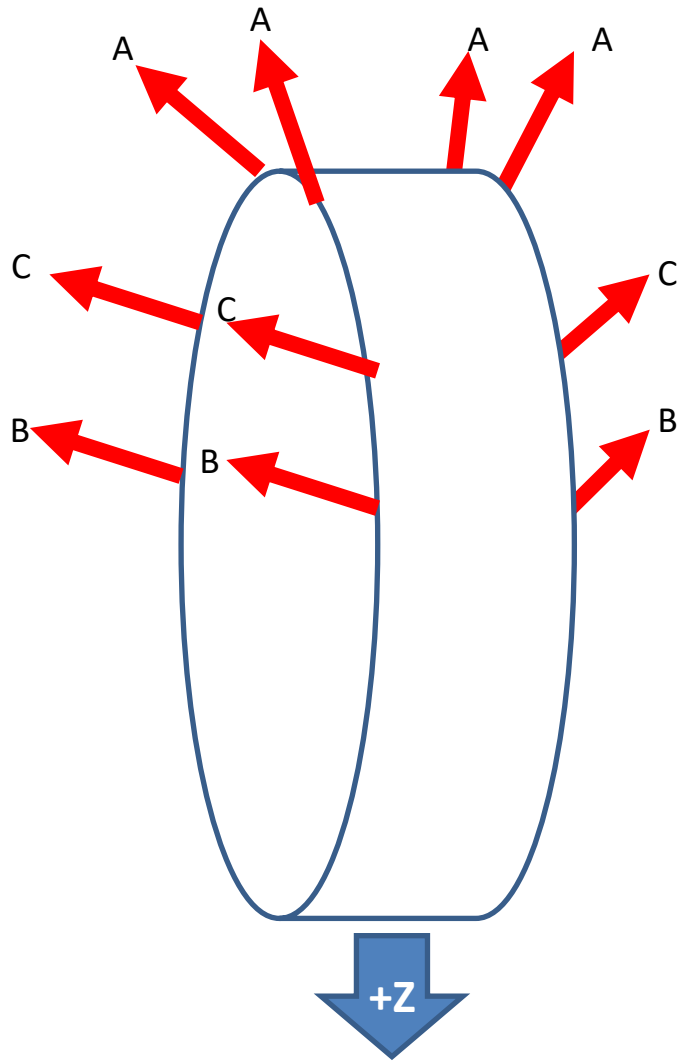
制御ロジック

-Z



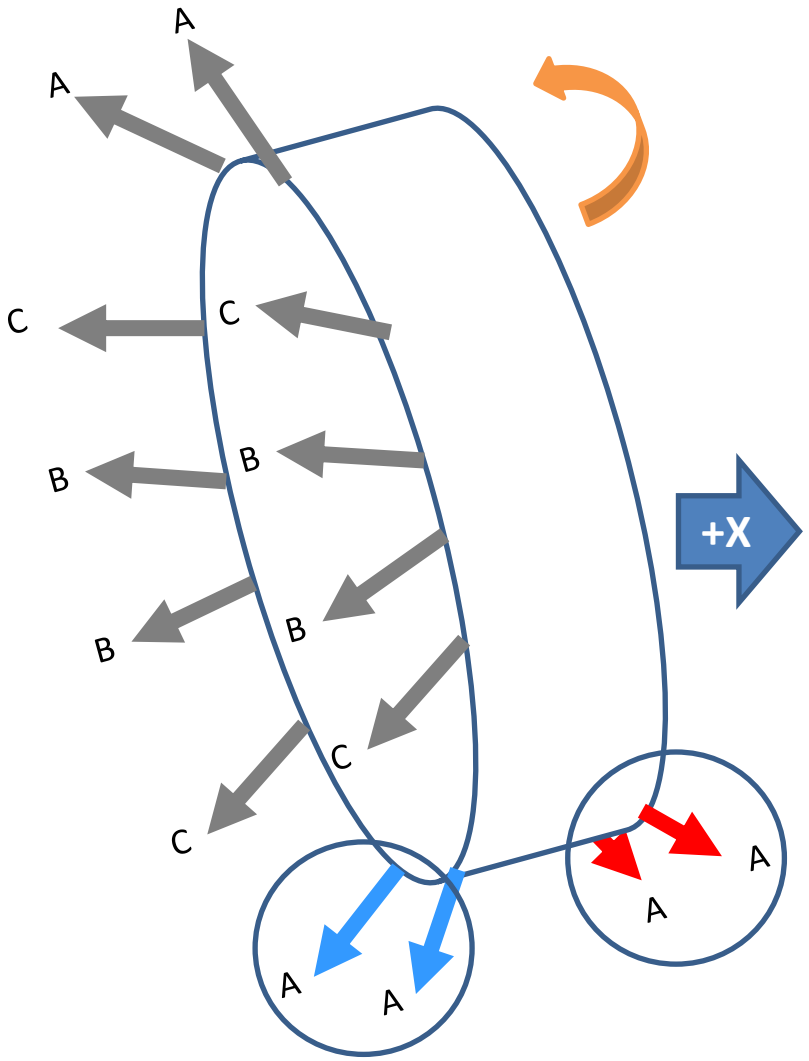
制御ロジック

+Z



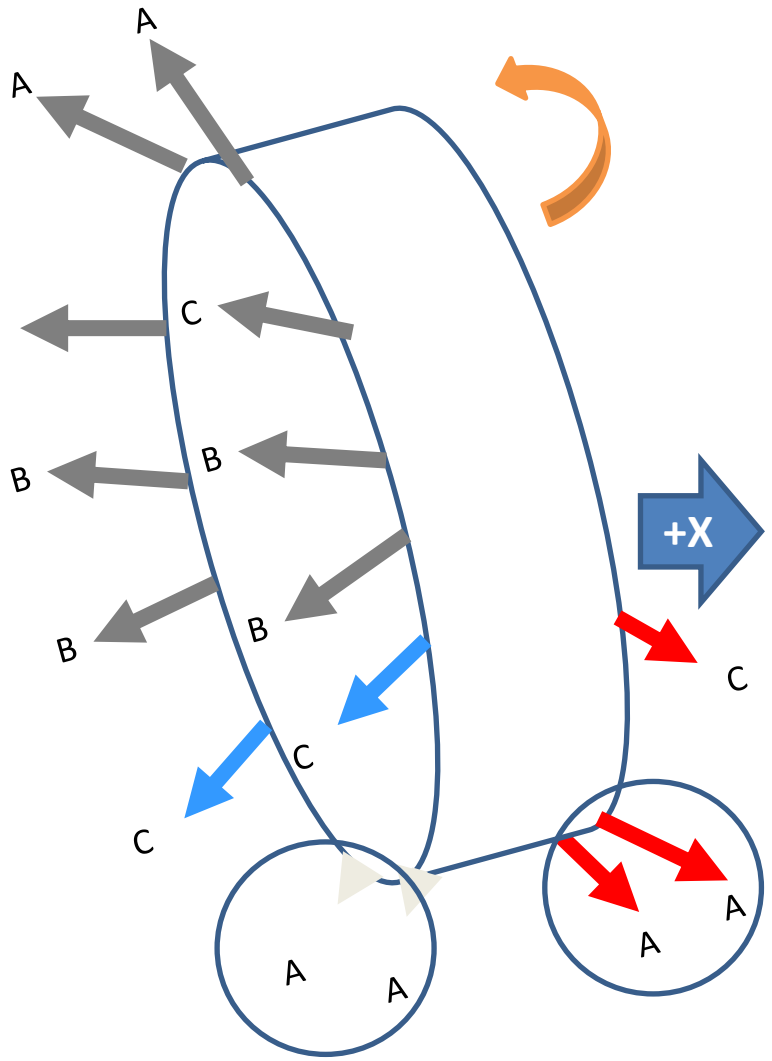
制御ロジック

スラスタ故障等による外乱が発生したら、その軸の制御に最も寄与の高い系のデューティを調整する



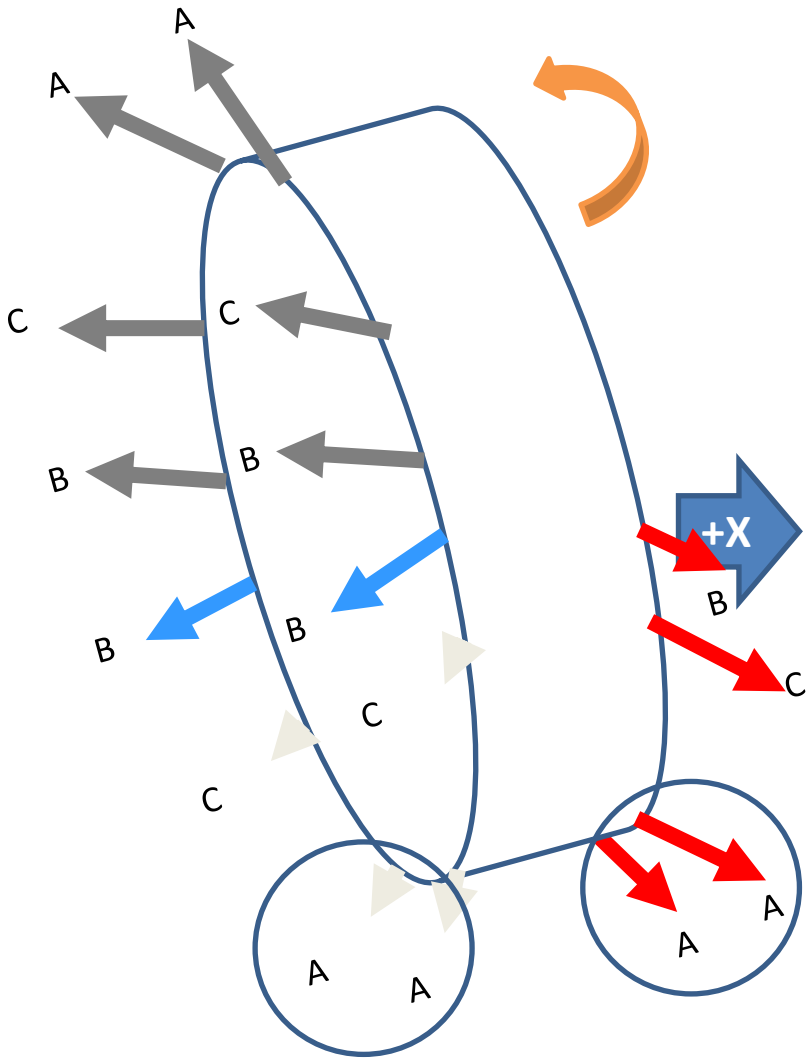
制御ロジック

一番寄与の高い系の調整で間に合わなければ、次にアサインされている系で調整



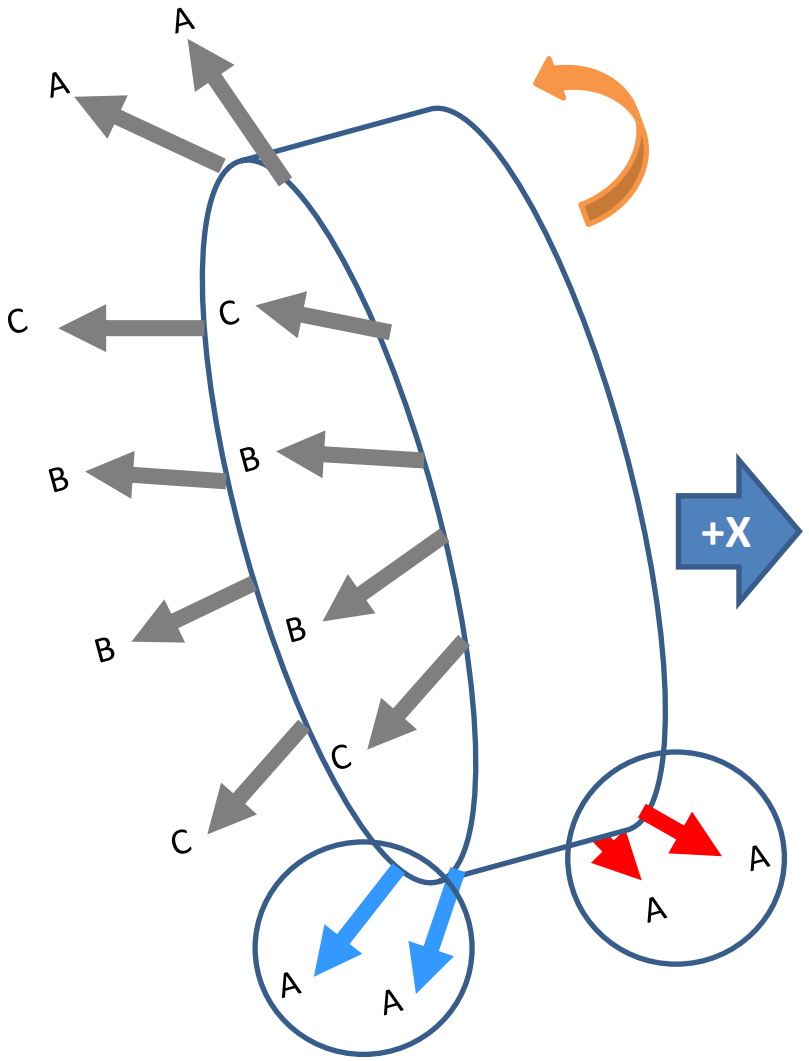
制御ロジック

それでも間に合わなければ全回転能力を使って外乱に対抗する



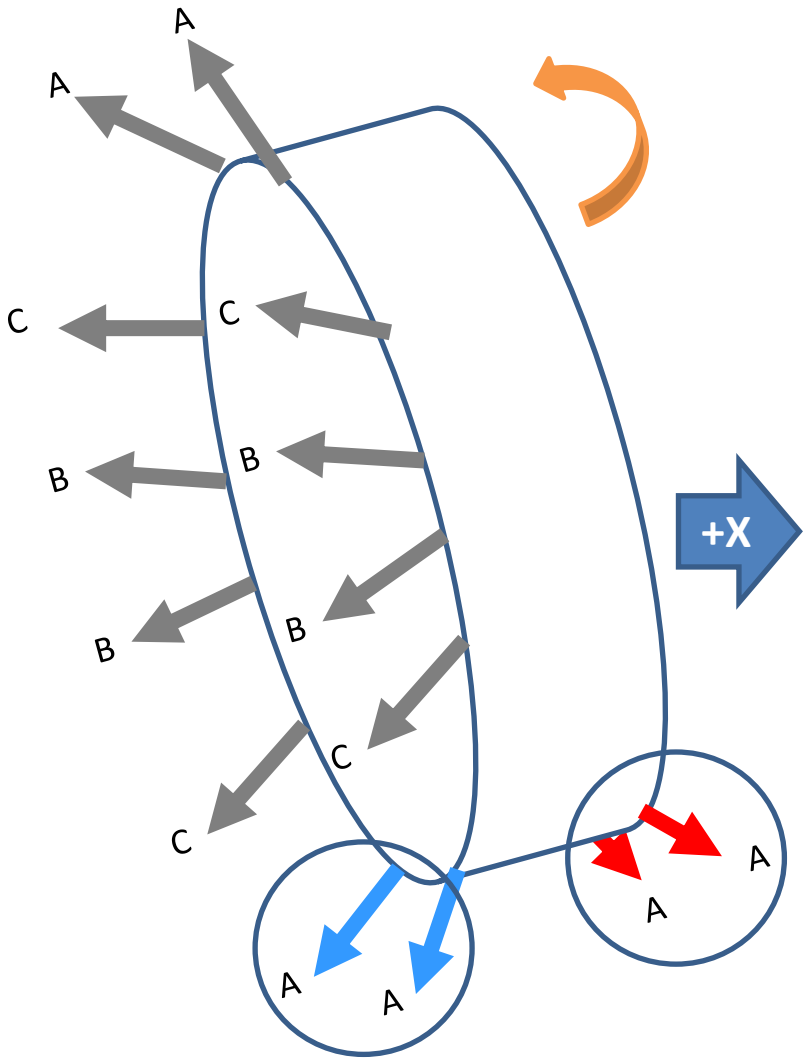
制御ロジック

シミュレーションの結果、最悪の外乱の組み合わせに対しても、１系のみの調整で十分（従来設計では切り替えが必要だった）



制御ロジック

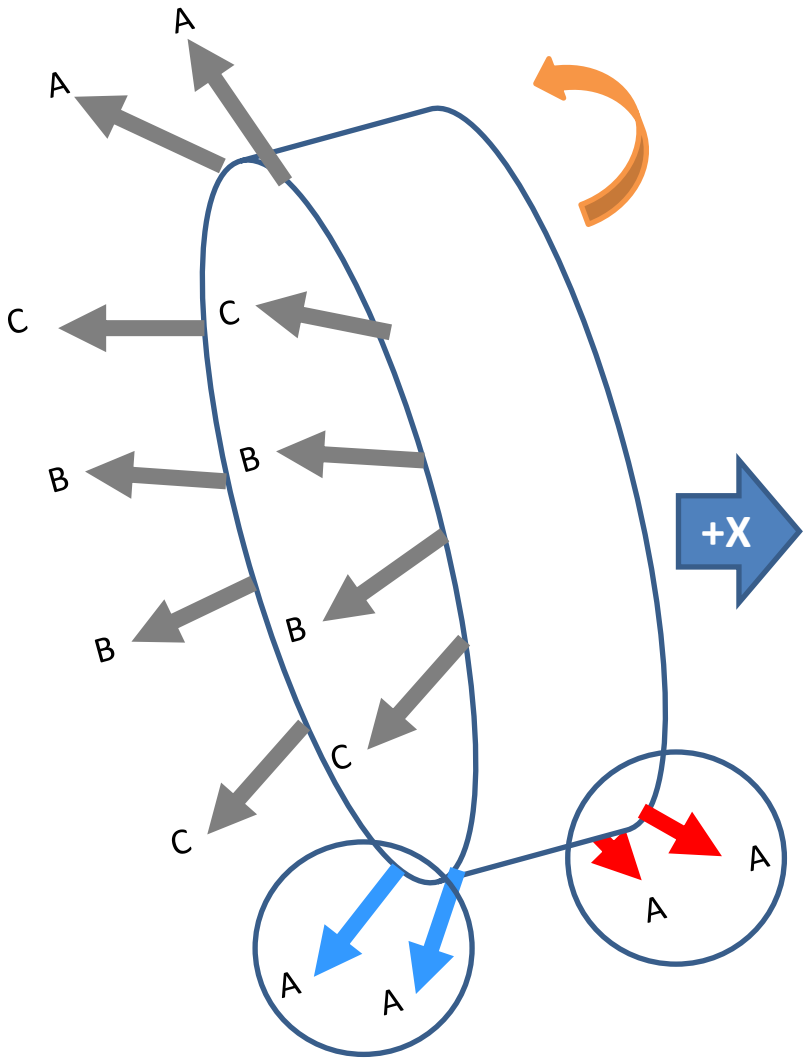
遺伝的アルゴリズムを使ったスラスター配置最適化を実施



制御ロジック

並進能力で5倍を達成
回転能力も大きなマージンを確保

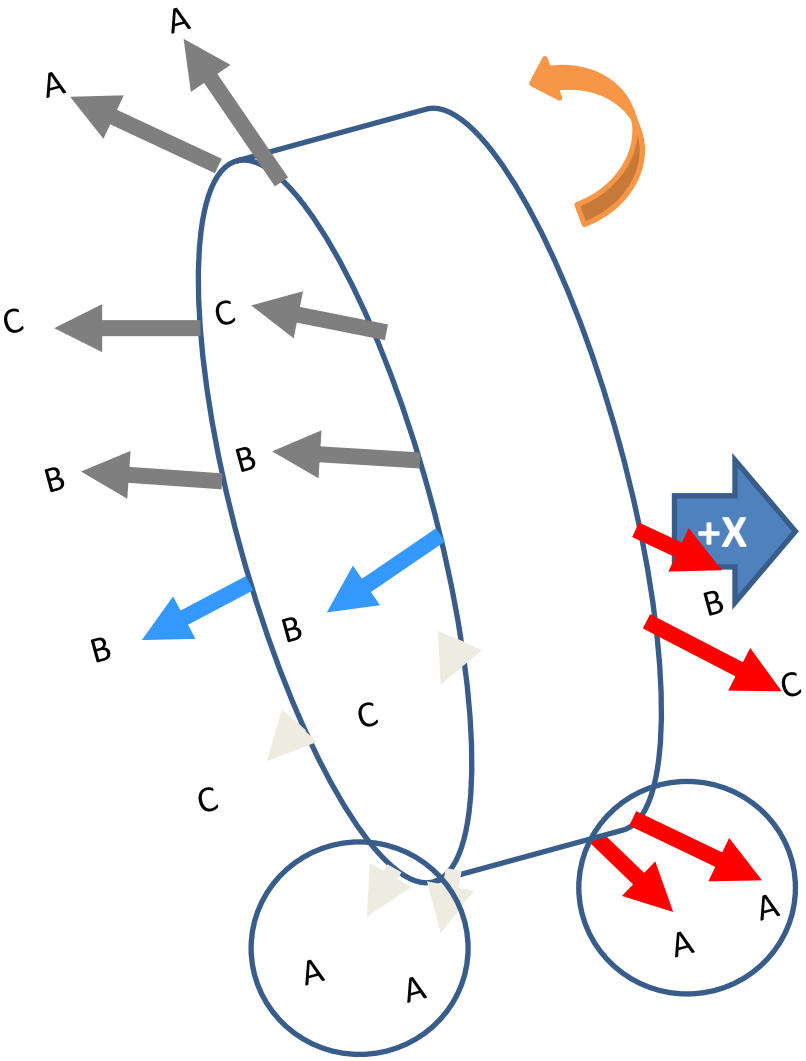
結果、メインエンジンを根こそぎ削除可能



制御ロジック

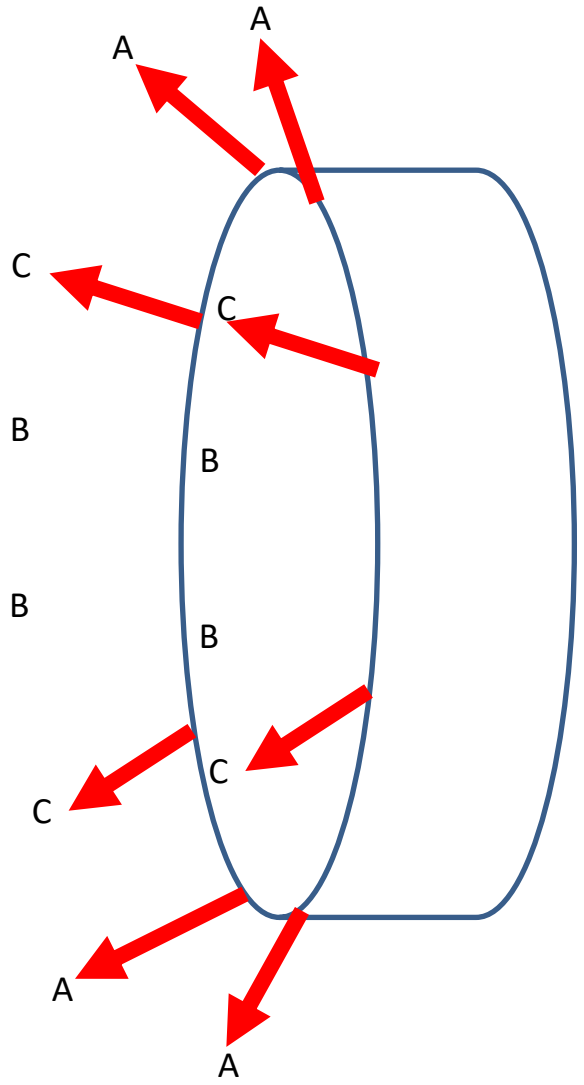
想定外の外乱に対する
マージンが豊富

持てる能力を最大限に
使えるから



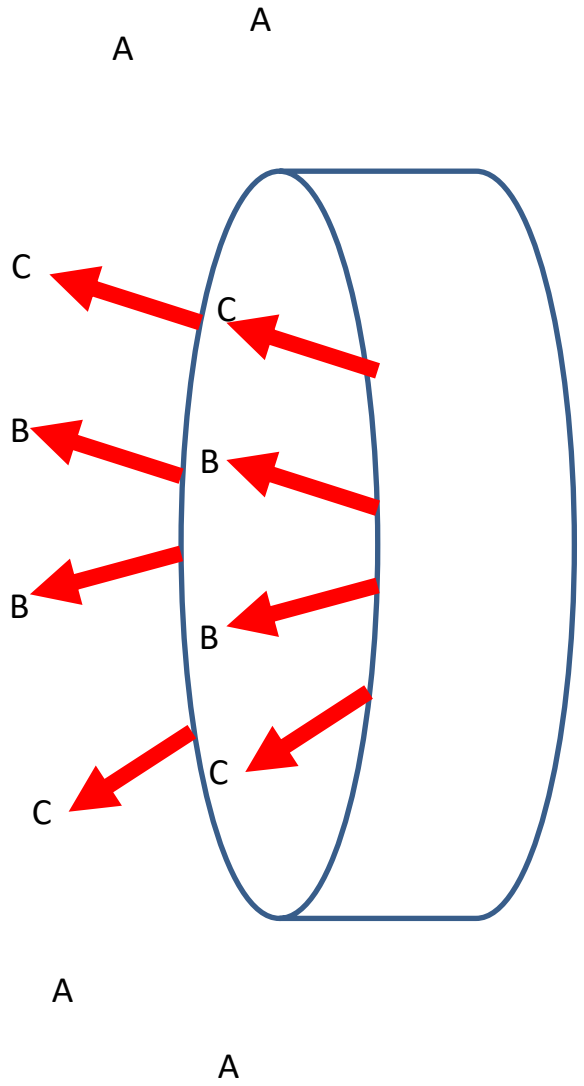
制御ロジック

特定の系を隔離（オフ）して、永久故障に対応したり、温存も可能



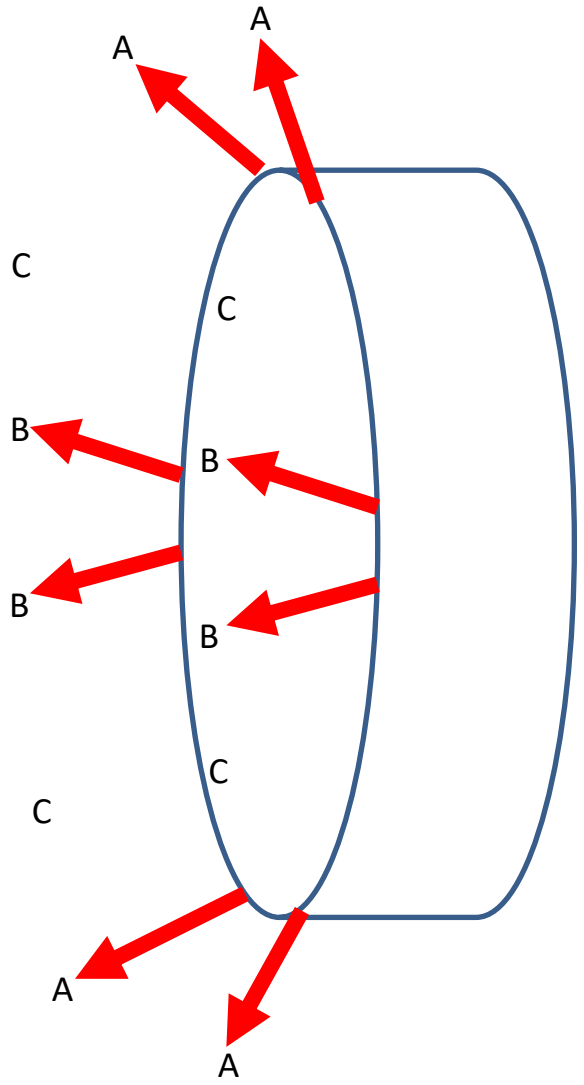
制御ロジック

特定の系を隔離（オフ）して、永久故障に対応したり、温存も可能



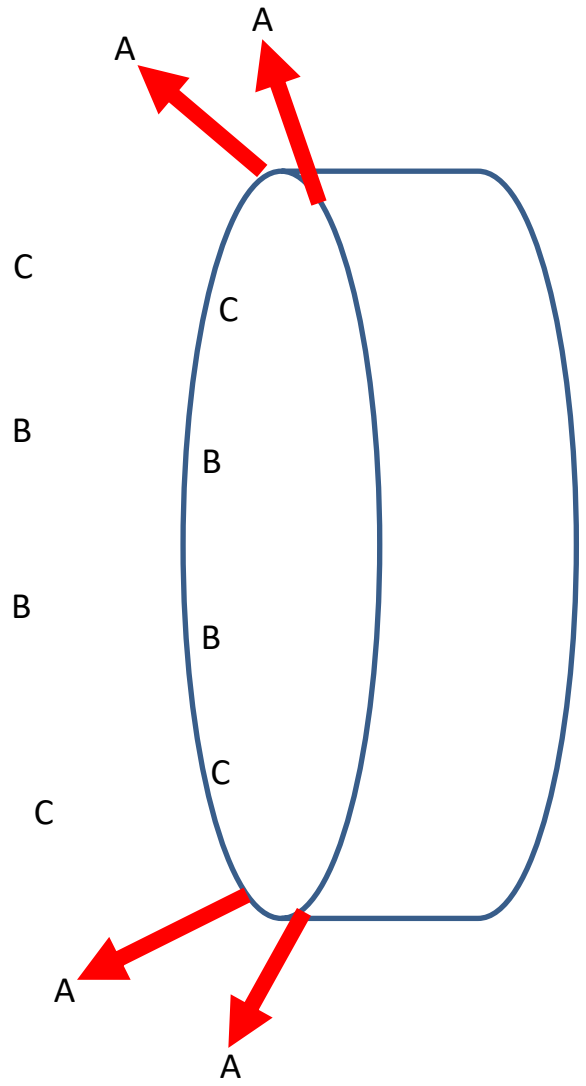
制御ロジック

特定の系を隔離（オフ）して、永久故障に対応したり、温存も可能



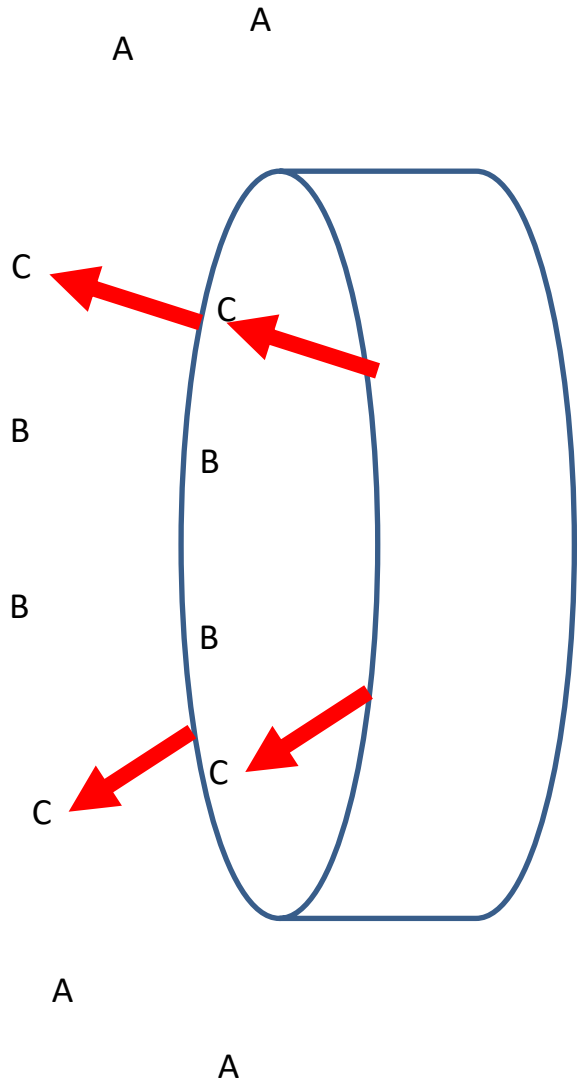
制御ロジック

特定の系を隔離（オフ）して、永久故障に対応したり、温存も可能



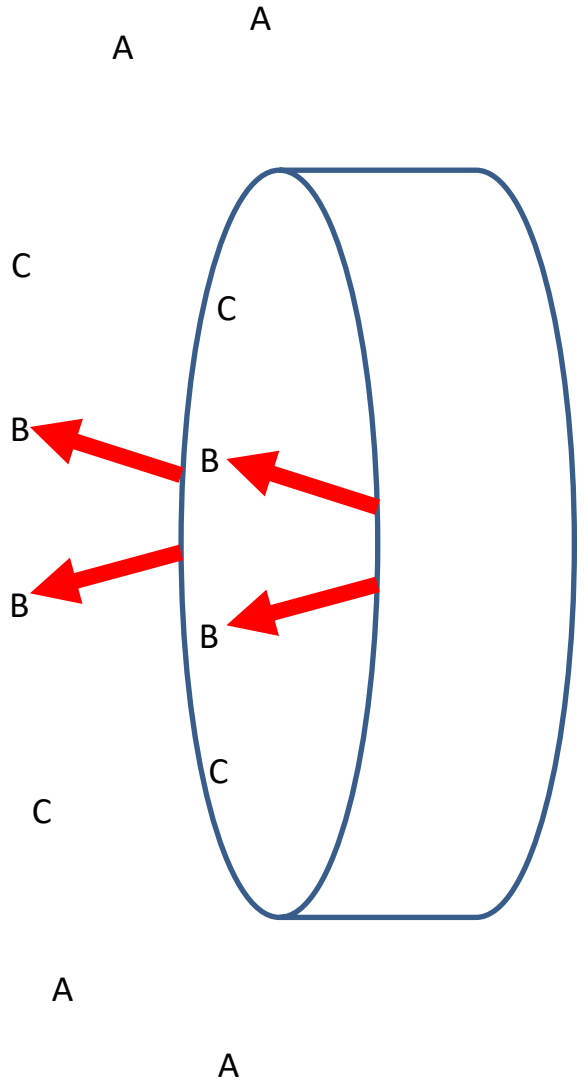
制御ロジック

特定の系を隔離（オフ）して、永久故障に対応したり、温存も可能



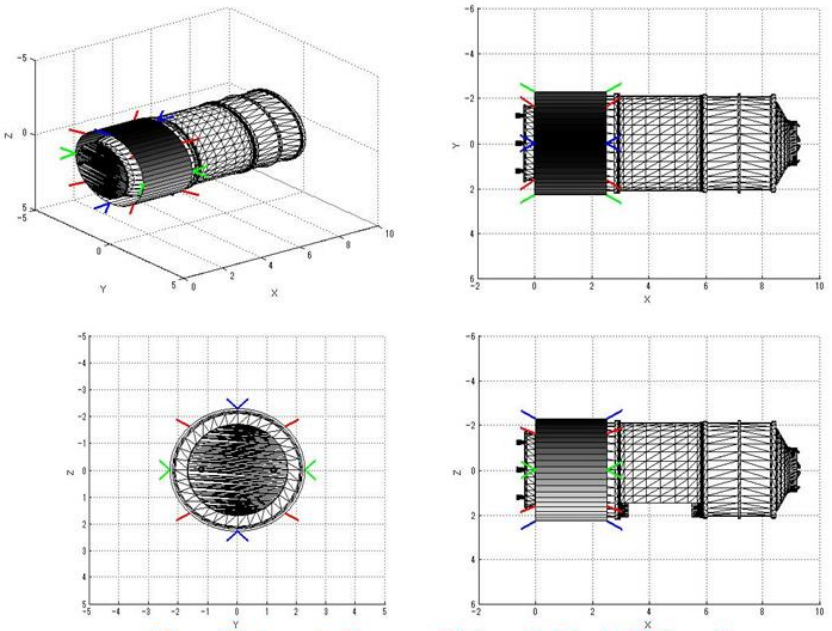
制御ロジック

特定の系を隔離（オフ）して、永久故障に対応したり、温存も可能



2. レジリエンス設計のコンセプト

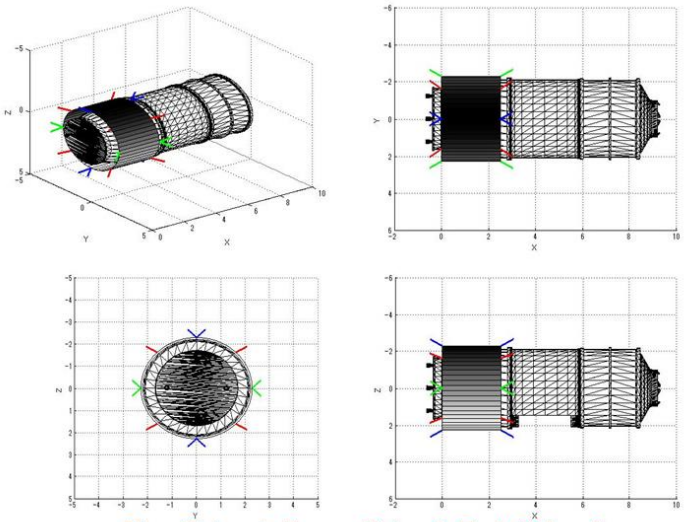
遺伝的アルゴリズムを用いた最適化
右図は、従来の衛星スラスター配置を模した
新型スラスターの配置である。
システムのモジュール化が目標となって
いたため、従来衛星のような、機体を
貫通するスラスター配管などは廃止
する前提で、最大限従来型に近い配置を
初期条件に設定し、遺伝的アルゴリズム
を実行した。



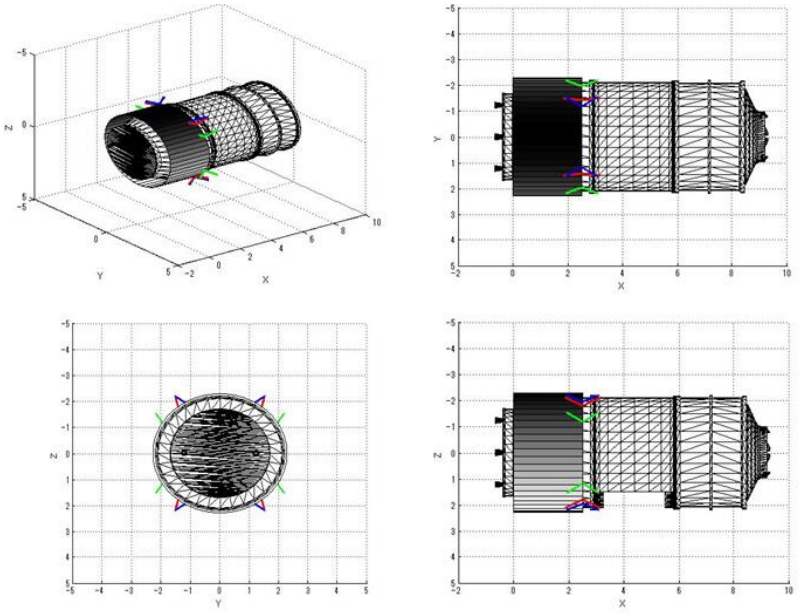
Blue: String-A, Green: String-B, Red: String-C

2. レジリエンス設計のコンセプト

遺伝的アルゴリズムを用いた最適化
遺伝的アルゴリズムが算出した最終的な最適解が右側である。

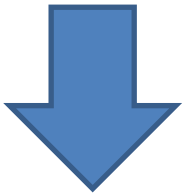


Blue: String-A, Green: String-B, Red: String-C



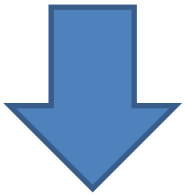
Blue: String-A, Green: String-B, Red: String-C

想定外の事象に対しても、柔軟に対応し、
しなやかに回復できる能力



レジリエンス

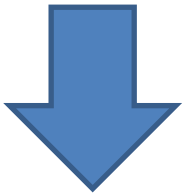
最良のパスを探す



レジリエンス

従来の考え：悪い箇所を見つけて切り替える

最良のパスを探す



レジリエンス

２系構成＝マージン2倍
３系構成＝マージン3倍
４系構成＝マージン4倍

.

.

安全になればなるほど
ミッション達成能力も上がる

ミッションと安全のトレードオフからの脱却

レジリエンス設計を導入した安全設計の特徴

従来の安全設計

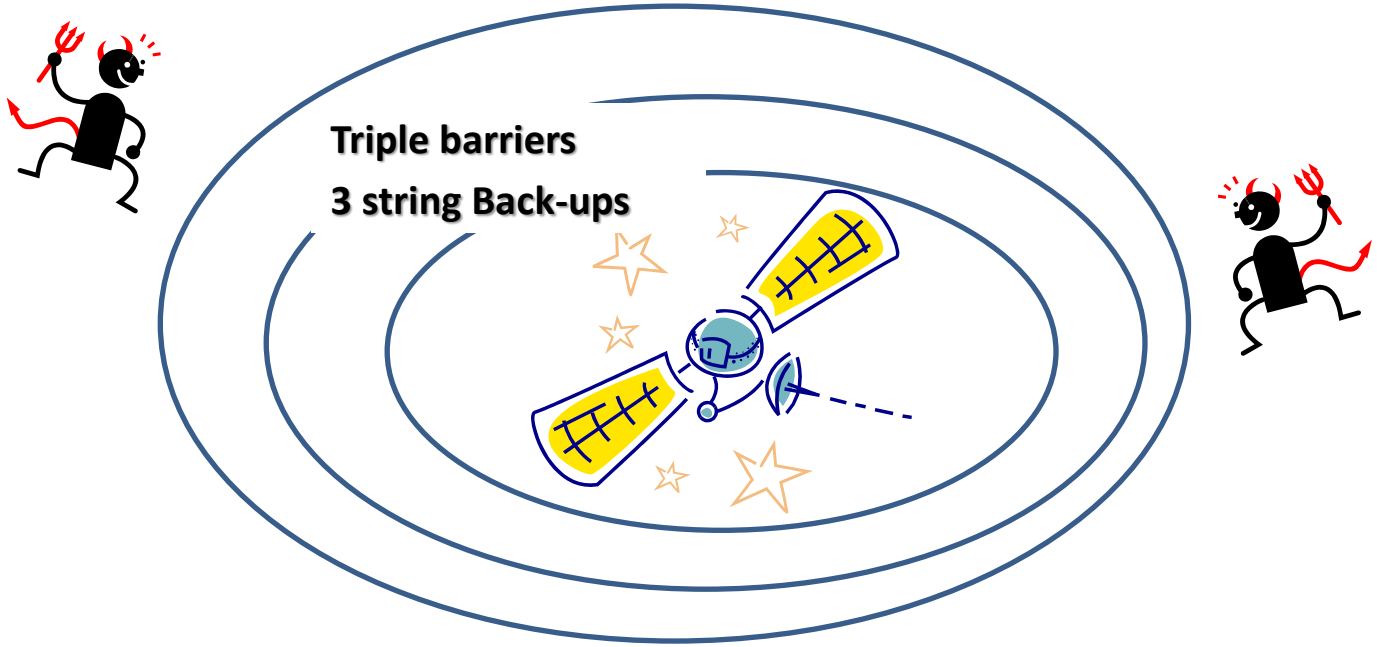
環境 = ばい菌

ばい菌の侵入を検知して、クリーンバックアップに切り替え

安全側の論理：システムは常にクリーンで完璧でなければならない

ミッション側の論理：クリーンでなくても飛び続けたい

常に両者はトレードオフの関係



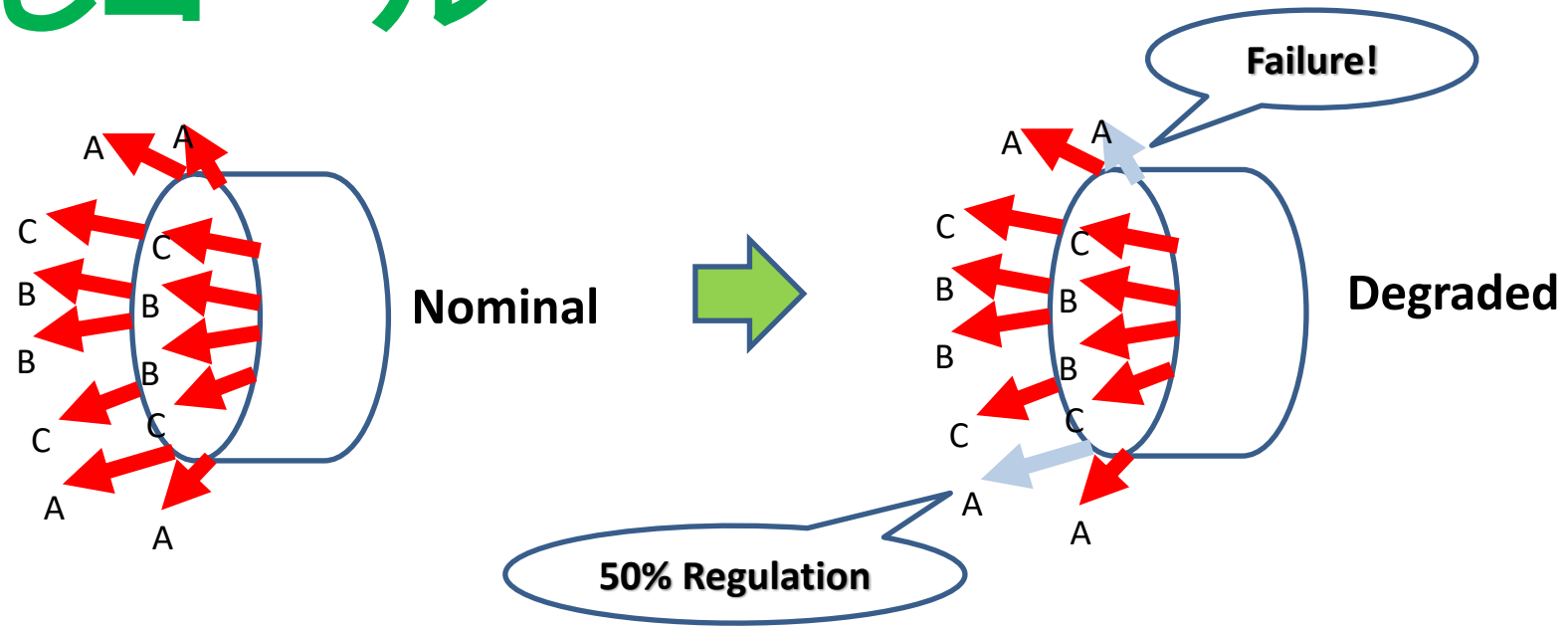
レジリエンスな安全設計

環境はばい菌ではなく、友達。デグレードしながら対応してゆくもの。

安全側の論理：クリーンでなくても飛び続けたい

ミッション側の論理：クリーンでなくても飛び続けたい

同じゴール



2つの安全:　従来の安全(Safety-I)
新しい安全(Safety-II)

"A Tale of Two Safeties"
Erik Hollnagel　2012

© www.erikhollnagel.com

Safety-I:　ばい菌に対抗するための**Protective**な安全

Safety-II: 最良の道を見つけるための Proactive**な安全**

2つの安全:　従来の安全(Safety-I)
新しい安全(Safety-II)

"A Tale of Two Safeties"
Erik Hollnagel　2012

© www.erikhollnagel.com

Safety-I:　常にクリーンなシステムを目指す。変化を嫌う。クリーンでなければアボート！

Safety-II: 環境に適応。変化を肯定。

2つの安全:　従来の安全(Safety-I)
新しい安全(Safety-II)

"A Tale of Two Safeties"
Erik Hollnagel　2012

© www.erikhollnagel.com

Safety-I:　安全は重荷（使われないバックアップ資源）

Safety-II: 安全はパワー（エクストラの制御能力）

従来の「安全」の定義

安全　＝　リスクの不在

新たな「安全」の定義

安全　＝　~~リスク~~の不在

安全　＝

変化する環境に

対応できる能力