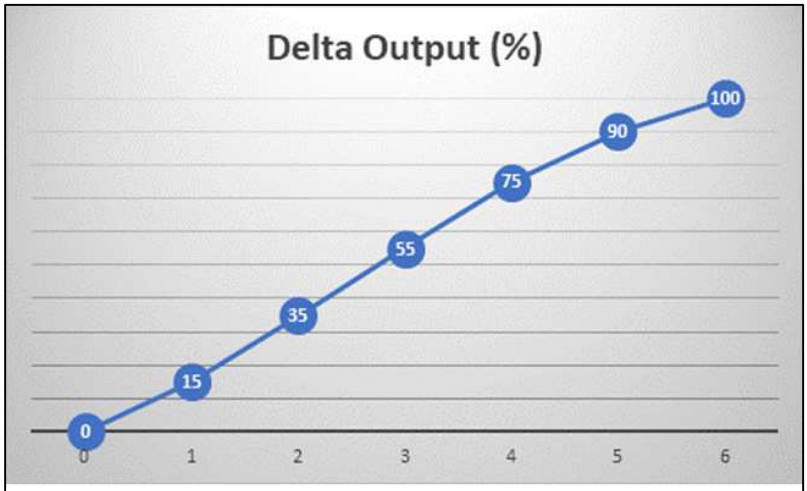
プロセスダイナミクス
第４回　SISO（Single-Input Single-Output）モデル予測制御

はじめに
プロセスダイナミクスとは、プロセスが運転中に示す挙動の事である。もしプロセスの挙動が安定であれば、そのままの運転を継続すれば良い。しかしプロセスの挙動が不安定であるなら、その不安定性の要因を見出し・修正することで、プロセスの挙動を安定な状態に改善する必要がある。
そのような観点から、前回までの３回で、プロセス運転の不安定性について、①クラスタリング技術を適用して所定の Tag 群のデータ解析から不安定性を見つける、②ウエーブレット変換及びフラクタル次元の解析から個々の Tag データの周波数をベースにした解析により不安定性を見つける、③新たに開発した“PSI（Process Stability Indicator)”と“HFSI（High Frequency Stability Indicator)”によって安定性/不安定性を定量化する、など事例を含めて説明した。
今回は、プロセス運転上の不安定性要因を取り除き、プロセス運転が安定した状態になることを前提として、その先にプロセス動特性で考慮すべきである“制御”について説明したい。ただし、Conventional な PID 制御ではなく、インパルス応答モデルを使った SISO（Single-Input Single-Output）モデル予測制御について説明する。

１．離散型コンボリューション・モデル（Convolution Model）の考え方
プロセス運転で使われるモデル予測制御は、離散型コンボリューション・モデルを基本形として構築される。
ここで、離散型コンボリューション・モデルを説明する前に、安定な運転を継続しているプロセスの制御について考えてみよう。
例えば、入力を増や（減ら）せば出力も増え（減）る制御系があるとする。しかし、この制御系への入力を増減させると、すぐに入力に見合った出力の変化が現れるのかというと、そうではない。入力の変化に対して、出力に係るバルブはゆっくり変化する。一般的にプロセス産業では、入力を変更したとき出力が瞬時に変化する制御系は存在しない。
だから、入力を１（単位）変化させたときの出力の変化は、例えば、１分後に最終出力の１５％、２分後に３５％、３分後に５５％、４分後に７５％、５分後に９０％、そして６分後に１００％になるように推移していく。（図-１）



図－１　入力に対する出力の変化（ステップ応答）

このグラフは、入力の１ステップに対する出力変化を表しているので、この関係はステップ応答と呼ばれる。

この「時間に伴う変化幅は、線形性が確保されたプロセス運転の制御系で常に一定である。」というのが、離散型コンボリューション・モデルの前提となっている。

もちろんこの関係が成立しない状況もある。例えば、制御系のバルブ開度が２０％以下または８０％以上である場合は入力を変化させても出力はほとんど変化しない、などである。

だから、この入力-出力変化の関係を把握する際は、少なくとも制御系のバルブ開度が３０－７０％であることが重要だ。そして、その際に把握した入力-出力変化の関係は、このバルブ開度内でのみ成立すると考える必要がある。

次に、制御系が安定でバルブ開度が３０～７０％であり、この入力-出力変化の関係が十分成立する場合の出力変化について考える。

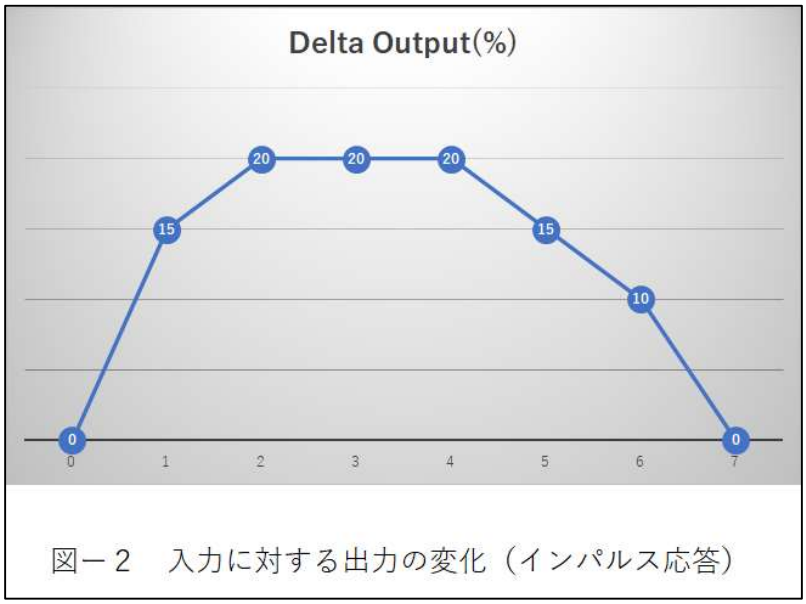
ある時間（ｔ）での出力変化ΔY(t)は、入力変化ΔX(t-1)～ΔX(t-６)により、以下の式で表される。

$$\Delta Y(t)=0.15\Delta X(t-1)+0.20\Delta X(t-2)+0.20\Delta X(t-3)+0.20\Delta X(t-4)+0.15\Delta X(t-5)+0.10\Delta X(t-6)$$

（係数は、％の絶対値ではなく、前回からの％の変化値。前述のステップ応答から差を計算したもの。）

この式の様に、出力変化ΔY(t)を過去の入力変化の重ね合わせで表現したモデルを、離散型コンボリューション・モデル（畳み込みモデル）と呼ぶ。

また、ここで使った係数の様に、出力変化を前回値からの変化幅で表した関係を、インパルス応答と呼ぶ。（図-２）



図－２　入力に対する出力の変化（インパルス応答）

ここで説明した関係を一般化した式で表現すると、出力変化（ΔY）は入力変化（ΔX）のコンボリューションの関数となる。

$\Delta Y(t)=\Sigma\ \alpha_i \Delta X(t-i)$

（パラメータの説明は、次の項を参照のこと）

２．入力変更値ΔX(t)を求める

SISO（Single-Input Single-Output）のモデル予測制御を行うには、目的値（ターゲット）や過去の挙動から、次の入力変更値ΔX(t)を求めればよい。

それは、以下のような計算から求められる。

（ここでは、５個のインパルス応答から成り立つ制御系であると仮定した。）

$\Delta Y(t) = a^*\Delta X(t-1) + b^*\Delta X(t-2)+c^*\Delta X(t-3)+d^*\Delta X(t-4)+e^*\Delta X(t-5)\cdots\cdots\cdots(1)$

また、最終ターゲット YTarget と現時点の出力値 Y(t)との関係は、以下の式になる。

$YTarget-Y(t)= a^*\Delta X(t) + b^*\Delta X(t-1)+c^*\Delta X(t-2)+d^*\Delta X(t-3)+e^*\Delta X(t-4)\cdots\cdots(2)$

（１）と（２）式で、

$\Delta Y(t)=Y(t)-Y(t-1)$

$\Delta X(t)=X(t)-X(t-1)$

$\Delta X(t-1)=X(t-1)-X(t-2)$

$\Delta X(t-2)=X(t-2)-X(t-3)$

$\Delta X(t-3)=X(t-3)-X(t-4)$

$\Delta X(t-4)=X(t-4)-X(t-5)$

$\Delta X(t-5)=X(t-5)-X(t-6)$

である。

ここで、(2)式から(1)式を引くと

YTarget-2*Y(t)+Y(t-1)　=a*ΔX(t)+b*X(t-1)-b*X(t-2)+c*X(t-2)-c*X(t-3)
+d*X(t-3)-d*X(t-4)+e*X(t-4)-e*T(t-5)
-a*X(t-1)+a*X(t-2)-b*X(t-2)+b*X(t-3)-c*X(t-3)+c*T(t-4)
-d*X(t-4)+d*X(t-5)-e*X(t-5)+e*X(t-6)
=a*ΔX(t)-(a-b)*X(t-1)-(b-c-a+b)*X(t-2)-(c-d-b+c)*X(t-3)
-(d-e-c-d)*X(t-4)-(e-d+e)*X(t-5)+e*X(t-6)

となる。これを単純に変換すると、

a*ΔX(t) = YTarget-2*Y(t)+Y(t-1)+(a-b)*X(t-1)+(b-c-a+b)*X(t-2)+(c-d-b+c)*X(t-3)
+(d-e-c+d)*X(t-4)+(e-d+e)*X(t-5)-e*X(t-6)
= YTarget-2*Y(t)+Y(t-1) +(a-b)*X(t-1)+(2b-c-a)*X(t-2)+(2c-d-b)*X(t-3)
+(2d-e-c)*X(t-4)+(2e-d)*X(t-5)-e*X(t-6)

となる。つまり、ΔX(t)は以下の式で表現できる。

ΔX(t)　=[YTarget-2*Y(t)+Y(t-1) +(a-b)*X(t-1)+(2b-c-a)*X(t-2)+(2c-d-b)*X(t-3)
+(2d-e-c)*X(t-4)+(2e-d)*X(t-5)-e*X(t-6)]/a

（パラメータ説明一覧）

YTarget　：Y 目標値
Y(t)　：制御変数、時刻 t における Y の値
ΔY(t)　：Y(t)-Y(t-1)
X(t)　：操作変数、時刻 t における X の値
ΔX(t)　：X(t)-X(t-1)
$\alpha, \beta, \gamma, \delta, \varepsilon$　：インパルス応答の標準的な係数
（$\alpha=0.1, \beta=0.3, \gamma=0.3, \delta=0.2, \varepsilon=0.1$）
Coef　：相関係数
RT　：リファレンストラジェクトリ（操作変数の動きを制限する変数）
a,b,c,d,e　：時系列の操作変数に対する係数
Dist(t)　：外乱、時刻 t における Dist の値

a= α *Coef
b= β * Coef
c= γ * Coef
d= δ * Coef
e= ε * Coef

ただし、一度の変化で Y(t)が YTarget と同一になる前提で計算しているので、ΔX(t)の変化が大きすぎる場合がある。そのために、RT（リファレンストラジェクトリ）という係数を使って、入力変化に制限を設ける。

X(t)=X(t-1)+ΔX(t)*RT

この X(t)が、次の入力値の設定値となる。
また、
ΔY(t+1) = a*ΔX(t) + b*ΔX(t-1)+c*ΔX(t-2)+d*ΔX(t-3)+e*ΔX(t-4)
ΔY(t+1)=Y(t+1)-Y(t)
だから、
Y(t+1)=Y(t)+ a*ΔX(t) + b*ΔX(t-1)+c*ΔX(t-2)+d*ΔX(t-3)+e*ΔX(t-4)
となり、もし制御変数に対する外乱 Dist(t+1)を考慮するなら、この式の最後に外乱の項を付ければよい。つまり、次の式となる。
Y(t+1)= Y(t)+ a*ΔX(t) + b*ΔX(t-1)+c*ΔX(t-2)+d*ΔX(t-3)+e*ΔX(t-4)+Dist(t+1)

３．SISO モデル予測制御の利点
前項で説明した SISO モデル予測制御は、最もよく使われている MIMO（Multi-Input Multi-Output）モデル予測制御と比べると、幾つかの点で利点がある。
それを説明する前に、MIMO モデル予測制御の利点と欠点について考えよう。
MIMO モデル予測制御には、以下の利点が考えられる。
１）ステップ応答テストから構築した各制御系の SISO モデルを組み合わせて行列式を作るので、個別の制御系の動特性は反映されている。
２）個々の制御系を調整する必要がないし、制御系間の干渉もある程度考慮されている。
また、以下の欠点も考えられる。
１）一つの行列式で複数の制御系への設定値変更を計算するため、各制御系に対する変更のインターバルを同じにしなければならない。プロセス産業で最もよく使われているのは、毎分の入力変更である。
２）制御目標に対し複数の制御系への設定値変更を行うため、個々の制御系の動特性がどのように反映されているか理解することは難しい。
これらの欠点をもう少し説明すると、
①　MIMO の制御群の中に、１分周期では遅すぎる制御系（例えば１０秒周期で動かす必要がある制御系）があったとしても、それに対応することは難しい。
②　制御がうまくいかない場合、個々の制御系のモデルがおかしいのか、制御系間の干渉がひどいのか、などの細かな検討が難しい。
ということである。
それに対して、２．で説明した SISO モデル予測制御は、
１）制御系のインターバルを個々に設定できる。
２）SISO モデルなので、制御がうまくいかない場合に容易に原因を特定できる。
３）制御系間の干渉は外乱として取り扱われるので、個々の制御系の RT を調整することで解消できる。
という利点がある。

我々の経験した例では、あるプロセスに４つの SISO モデル予測制御を導入したが、それぞれ１０秒、３０秒、１分、５分のインターバルで操作変数の設定値変更を行った。うまく動かない制御系のモデルは運転中に係数を変更して、また RT により干渉を排除することもできた。

４．Excel を使った SISO モデル予測制御のシミュレーション

今まで説明した SISO モデル予測制御は離散系なので、Excel を使って簡単にシミュレーションできる。

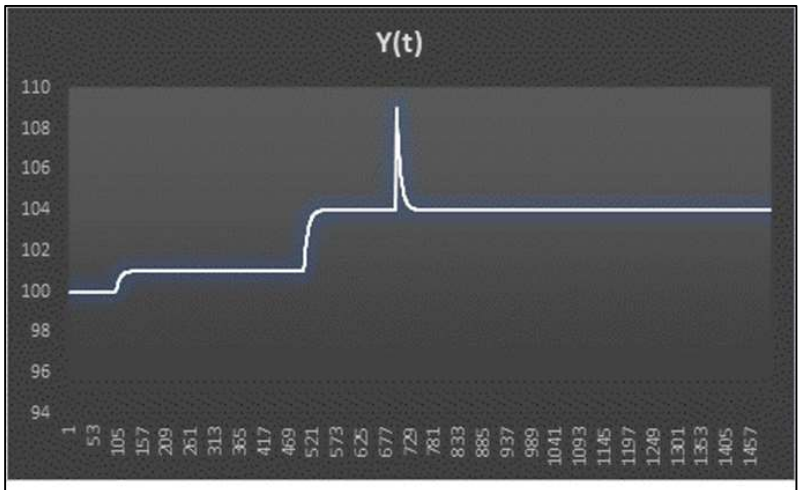
図-３の例では、B 列に各種係数などの条件、D 列に時間軸、F 列に YTarget、G 列に Y(t)、H 列に ΔY(t)、I 列に Dist (外乱)、J 列に X(t)、K 列に ΔX(t)、そして L 列に ΔX(t)real(ΔX(t)に RT をかけたもの、実際の入力変更値)を定義してある。

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	CPCon Simulation			Time		Ytarget	Y(t)	ΔY(t)	Dist	X(t)	ΔX(t)	ΔX(t)real	
2				1		100	100		0	30			
3	Project	TEST		2		100	100	0	0	30	0	0	
4	Date	2021.09.01		3		100	100	0	0	30	0	0	
5	Trial	1		4		100	100	0	0	30	0	0	
6				5		100	100	0	0	30	0	0	
7	α	0.1		6		100	100	0	0	30	0	0	
8	β	0.3		7		100	100	0	0	3.00000E+01	0.00000E+00	0.00000E+00	
9	γ	0.3		8		100	1.00000E+02	0	0	3.00000E+01	0.00000E+00	0.00000E+00	
10	δ	0.2		9		100	1.00000E+02	0	0	3.00000E+01	0.00000E+00	0.00000E+00	
11	ε	0.1		10		100	1.00000E+02	0	0	3.00000E+01	0.00000E+00	0.00000E+00	
12				11		100	1.00000E+02	0	0	3.00000E+01	0.00000E+00	0.00000E+00	
13	Coef	5		12		100	1.00000E+02	0	0	3.00000E+01	0.00000E+00	0.00000E+00	
14	RT	0.01		13		100	1.00000E+02	0	0	3.00000E+01	0.00000E+00	0.00000E+00	

図－３　ExcelをつかったSISOモデル予測制御シミュレーション

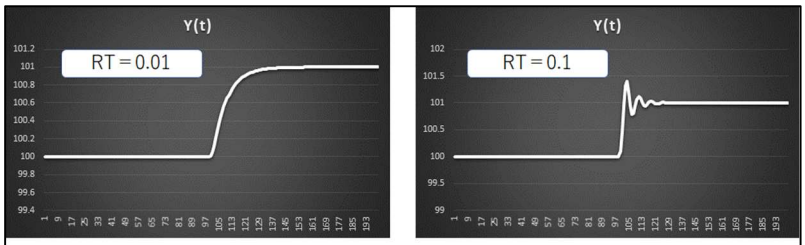
YTarget と Dist の値を変えることで、時系列のシミュレーションが可能である。また、適切な RT 値の決定にも使える。

シミュレーションで求めた Y(t)の例を、図-４に示した。これは、Ytarget を 100 から 101、次に 104 に変更し、その後一時的な外乱を加えて、応答性を確認したものである。目標値に対する追従性が良いことが分かる。



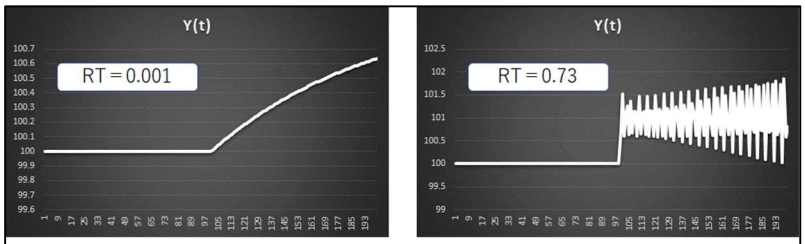
図－４　SISOモデル予測制御シミュレーション例

図-５は、RT 値による応答性の違いを示した。この制御系の例では、RT を 0.1 に変更すると、RT=0.01 の場合に比べ、オーバーシュートさせながら目的値に追従させていく結果となった。また、目的値にたどり着くまでの時間は、オーバーシュートさせた方が短いことも分かった。



図－５　RT値による応答性の違い

図-６には、RT 値が小さすぎる場合と大きすぎる場合の結果を示した。RT 値が小さすぎると左図のように不十分な応答になってしまい、RT 値が大きすぎると右図のように収束せずに発散してしまう。



図－６　RTが小さすぎる、大きすぎる場合の応答

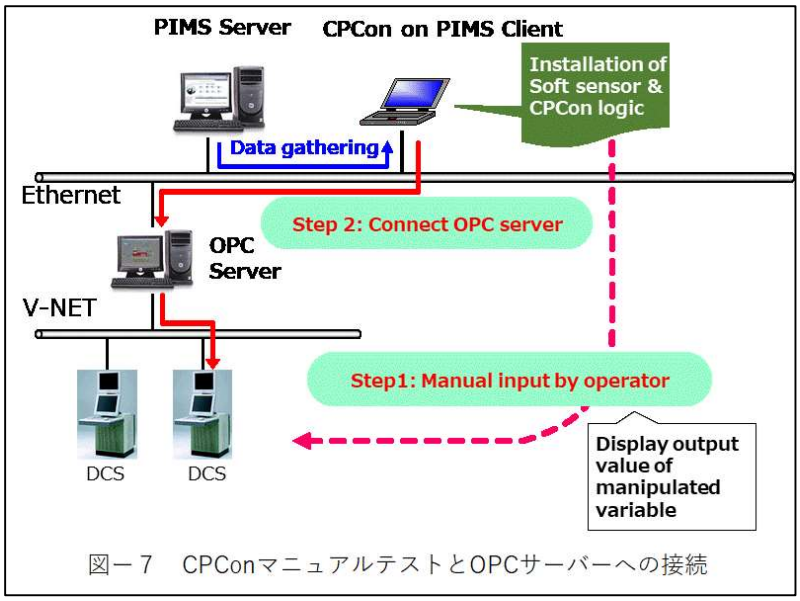
以上の例のように、このシミュレーションを使えば RT 値を含めた満足できる SISO モデル予測制御の検討が行えるので、新しく導入する制御系のリスクアセスメントの点からも、効果のある事前検討が可能となる。

５．SISO モデル予測制御のマニュアルテスト

ただし、新しく導入する制御系は、シミュレーションによる動作確認だけではリスク検討は不十分で、実際の接続・起動を行う前に実機での応答性を確認することが重要である。だが、一般的に導入されている MIMO モデル予測制御ツールでは、個々の制御系の応答性を DCS への接続なしに確認することは不可能である。

その欠点を補うために、CPCon による SISO モデル予測制御では、DCS への接続なしに実機での応答性を確認するマニュアルテストを実施する。

図-７は、CPCon システムの簡単な構成である。CPCon は、既存の PIMS から時系列データを入手して Clustering や Wavelet/Fractal による安定性解析、そして SISO モデル予測制御の計算を行う。



図－７　CPConマニュアルテストとOPCサーバーへの接続

SISO モデル予測制御で求められた操作変数の値（X(t)）は、CPCon の画面に表示されるので、その値を計算のインターバルに合わせて、オペレータが PV 値を手入力で変更し、制御性の応答を確認する。(Step-1：オペレータによるマニュアルテスト)

そして、その応答性が満足できるものであることを確認した上で、OPC サーバーと接続し、OPC サーバー経由で DCS に操作変数の新しい入力値を送信する。(Step-2：OPC サーバー接続)

これらのステップを踏むことで、更なるリスクアセスメントが可能となり、安心して制御系を運用できるのである。

６．OPC サーバー接続

CPCon は、全て Python 言語で開発されたアプリケーションである。SISO モデル予測制御も Python であり、アプリケーションから DCS へのデータ送信（操作変数の変更値）は OPC サーバーを経由して実行される。

このPythonプログラムからOPCサーバーにデータを送信するためには、Python-OpenOPC3Xモジュールのインストール及びデータ通信をハンドリングするDLLファイルの設定が必要となる。それらの設定した後、以下のプログラムで通信を行う。

```
opc_class = 'Graybox.OPC.DAWrapper'
client_name = 'OpenOPC'
opchost='（OPCサーバーのIPアドレス）'
opcserv='（OPCサーバー名）'
opc = OpenOPC.client(opc_class, client_name)
opc.connect(opcserv,opchost)
opc.write(('CPCNaphtha',mv_last))
opc.close()
```

この中のopc.writeで、OPCサーバー上の名前（ここではCPCNaphtha）と実際に送信する値（mv_last）を定義する。
それ以外の変更箇所は、OPCサーバーのIPアドレスとサーバー名である。
また、OPCサーバー上でファイアウォールを設定している場合には、そのPORTを開く必要がある。

おわりに

４回を通じて、「プロセスダイナミクスで最も重要なのは、プロセス運転が安定か不安定かを見極め、不安定であれば原因の特定と排除を行う。不安定性は、時系列での検討（Clustering）と周波数での検討（Wavelet/Fractal）を行う必要がある。プロセス運転が不安定であれば、SISOモデル予測制御を導入することで、満足できる制御系を構築できる。その際、制御系の安定性・追従性について、事前に検討が必要である。」について説明した。
また、そのために我々が開発したCPConシステムについても簡単に触れた。
最後に、まとめとして、一般的なMIMOモデル予測制御システムとCPConシステムについて比較する。
一般的なMIMOモデル予測制御の構築では、運転の安定性についての検討はあまり実施されない。個々の制御系に対するPIDチューニングは実施するが、それ以上の安定性解析は適用されない。
今回は説明を省いたが、品質制御のための品質推定式は、基本的にはPCT（圧力補正を付加した抜出温度）による推定式であり、実際の品質と推定値が異なる場合が多い。
必要な制御群を一つのマトリクスで処理するために、多くの相互干渉が発生する。相互干渉は、コスト関数を使った重みづけで最小化が図られるが、運転条件によってコスト関数が変わると機能しない。
制御のための計算は毎分実行されるため、数秒から数十秒ごとに制御が必要な制御系が含まれると機能しない。

制御システムは、DCS に直接接続されるために、装置運転中の導入は不可能である。接続するためには、一度 DCS を停止する必要がある。
アプリケーションは、新しく設置するコンピュータに導入する必要がある。
制御系のテストは、全ての導入が完了した後に行われる。事前にそれぞれの制御系を確認することは、不可能である。
それに対し CPCon のアプローチでは、運転の安定性を重視する。そのため、運転データの解析に対し、Clustering による時系列の安定性解析と、Wavelet/Fractal による周波数帯での安定性解析を実施し、その結果は PSI（プロセス安定性指数）や HFSI（高周波帯安定性指数）として定量化され、その値によって安定か不安定か自動的に評価される。
確認された運転不安定箇所についてプロセスエンジニアリングを実施し、運転条件の変更や PID チューニングなどによる安定化対策を実施する。
それでも安定化されない箇所について、CPCon 安定化制御を導入する。
安定化制御は、必要な箇所に適用されるため、複雑なマトリクスによる制御は必要ない。またコスト関数による重みづけも必要ない。
CPCon の品質推定式は、AI 技術（Regression）を利用した推定式であり、PCT による推定に比べて推定精度が高い。この精度の高い推定式を用いて、CPCon 品質制御を構築する。
CPCon では、それぞれの制御系に異なった制御インターバルを設定できる。数秒から数十秒ごとの制御も可能である。
制御システムは、既存のコンピュータ上に導入される。新しいコンピュータは必要ない。
制御系システムと DCS の接続は、OPC サーバーを経由して行われる。
システム導入の際に必要なのは OPC サーバーとの接続であり、DCS 運転中でも問題なく導入できる。また、それぞれの制御系は、OPC サーバーと接続する前に Manual Input で性能を確認できる。

マトリクスを使った MIMO モデル予測制御の普及によって、高度制御はベンダーが開発するものになってしまった。モデルは、ステップ応答テスト期間中に設定した運転条件に適応するように定義され、運転条件がそれから大きく変化した場合は、再度ベンダーにモデル修正を依頼しなければならない。自分たちの装置の運転に対する挙動（プロセスダイナミクス）を理解せずに、ベンダーから与えられたモデル予測制御を使うことになってしまった。
だが、それでいいのだろうか？
我々は経験から、プロセスエンジニアはプロセスダイナミクスを十分理解する事が重要である、と考えている。プロセスエンジニアは、プロセス運転の挙動を理解することで、異常をいち早く検知できる。そしてそれが、トラブル発生の芽を摘む事に繋がる。
だから、（最新技術を使って）プロセス運転の安定性/不安定性を、自らの手で解析することが重要である。
安定であればそのまま運転を継続すれば良い。不安定なら、原因を特定して不安定性を改善

する。その時、プロセスエンジニアはプロセスプラントに対する理解度がアップする。また、自分で制御系を設計することで、対象となるプロセスのダイナミクスも熟知できる。
ベンダーに開発・導入を任せていたのでは、このようなエンジニアリング技術の向上、プロセスダイナミクスに対する洞察力の向上、は望めない。
プロセスプラントは、プロセスエンジニアに、もっとプロセスダイナミクスを理解してもらい、より効率的な運転を行ってもらいたい、と期待しているのではないだろうか。