フィルム搬送および巻取りにおける速度と張力の制御

西田　健
九州工業大学大学院　工学研究院
機械知能工学研究系　知能制御工学部門

Contents

1. はじめに

ウェブとは

薄く長い素材の一般名称

紙・板紙，繊維，プラスティックフィルム，セラミックシート，
炭素繊維複合材，不織布，合成紙，金属箔，鋼板，
液晶パネル用各種光学フィルム，固形高分子膜，
医療用人工生体膜

ウェブの製造技術

コンバーティング技術

コーティング，ラミネート，プリンティングなど，学術的に目覚ましく発展している．

ウェブハンドリング技術

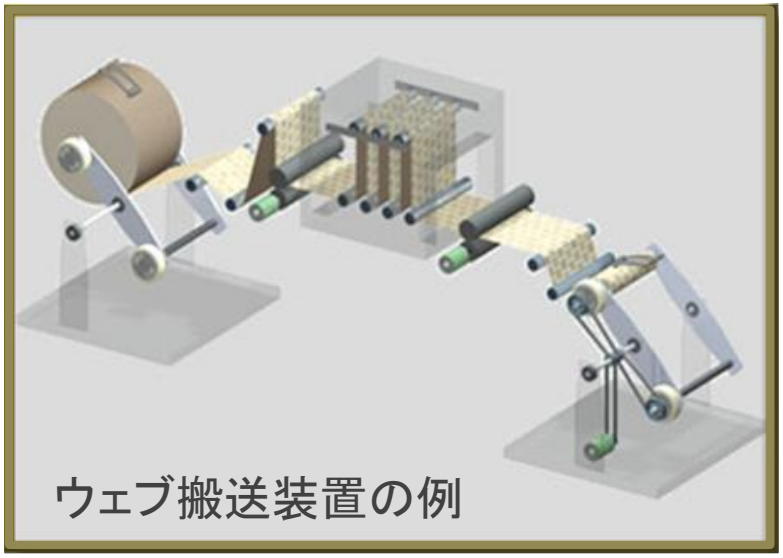
- ウェブの巻出し，搬送，巻き取りを行う技術
- 生産現場の経験の積み重ねにより練り上げられてきた

1. ウェブ搬送および巻取りシステムの重複分割分散制御

ウェブ搬送装置

ウェブ搬送装置は，多数の駆動ローラを制御することによって，ウェブの速度や張力を高い精度で一定に保つ．

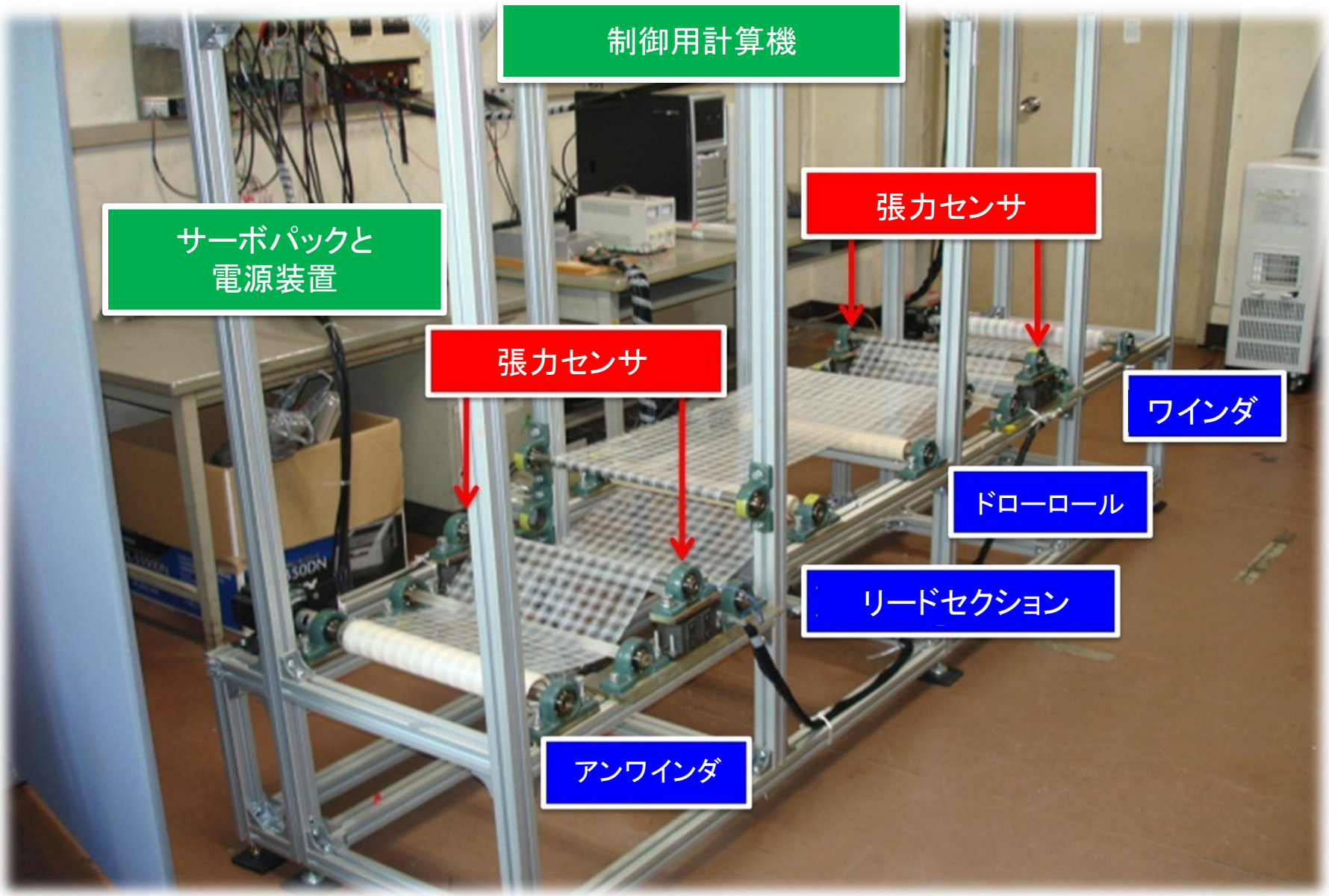
- 破損や破断，巻き取りムラを抑制
- 安定した品質の維持


ウェブ搬送装置の例

制御を困難にする要素

1. 一般にウェブ搬送系は大規模であるため，分散制御系が構成される．
   - サブシステム間に相互干渉が発生
   - 外乱やパラメータ変動に脆弱
2. 時変かつ非線形なダイナミクスを有する．
3. 精密なモデリングやエミュレートが困難．

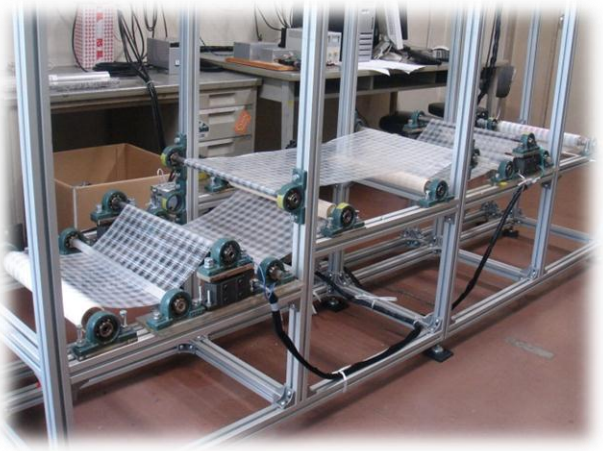
1-1. ウェブ搬送系のモデル化



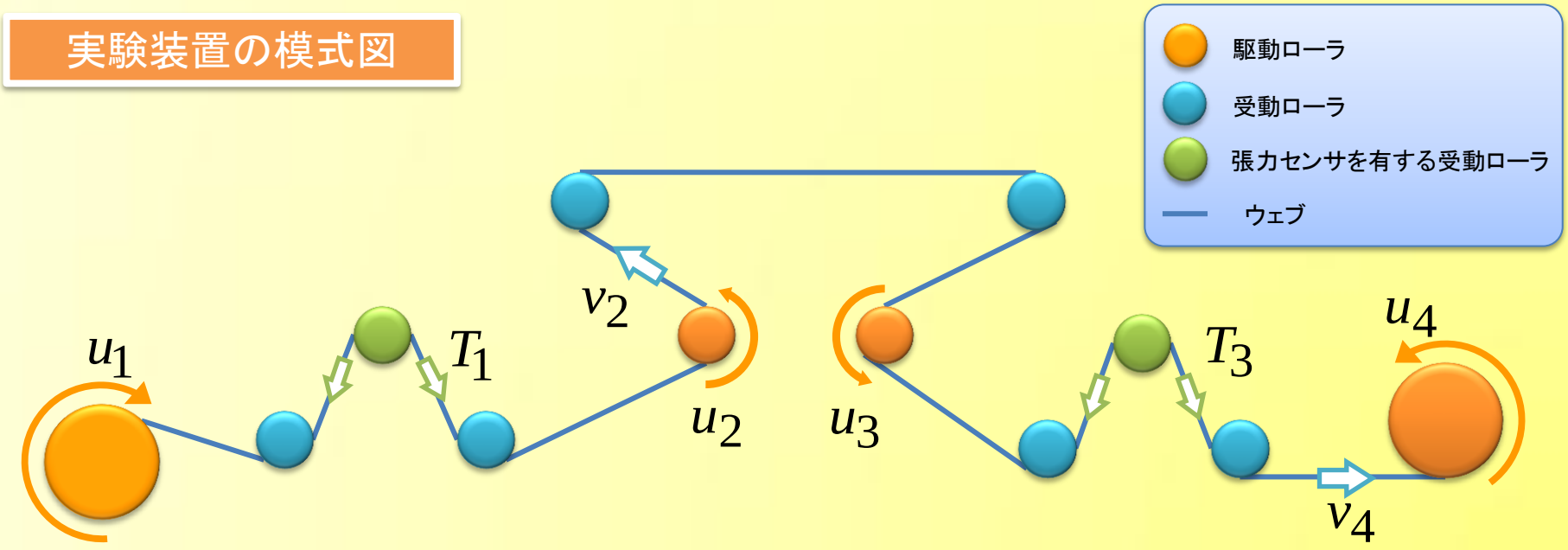
1-1. ウェブ搬送系のモデル化

ウェブ(フィルム)搬送システムの実験装置

- 制御周期　10[ms]
- サーボモータ4台，張力センサ4台
- ローラ12本
- 500[mm](W) x 2500[mm](L) x 1500[mm](H)



実験装置の模式図



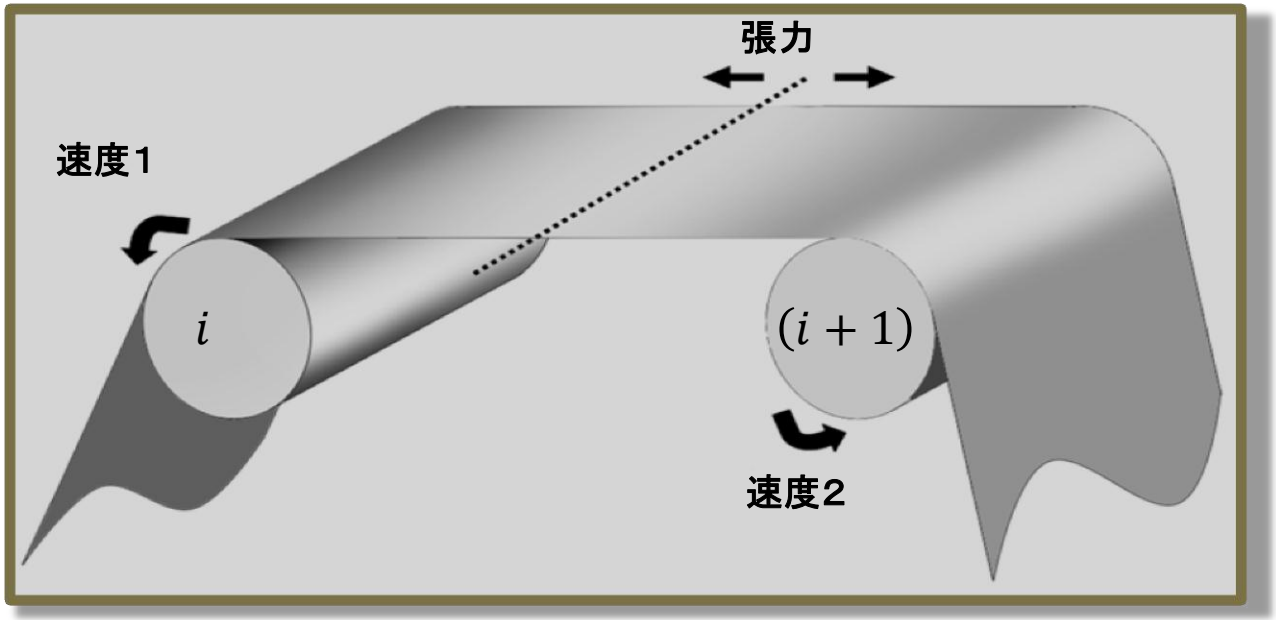
1-1. ウェブ搬送系のモデル化

ウェブおよびシステムのシステムパラメータと変数

A [m^2]	ウェブの断面積
G_v [$\mathrm{N/m}^2$]	ウェブ材料の弾性率
η_v [$\mathrm{Ns/m}^2$]	ウェブ材料の粘性率
r_i [m]	第i駆動ロールの半径
J_i [kgm^2]	第i駆動ロールの慣性モーメント
L_i [m]	第iと第$(i+1)$駆動ロール間のウェブの長さ
u_i [Nm]	第i駆動ロールの入力トルク
v_i [m/s]	第i駆動ロールにおけるウェブ速度
T_i [N]	第iと第$(i+1)$駆動ロール間のウェブ張力

1-1. ウェブ搬送系のモデル化

ウェブの張力と速度の関係



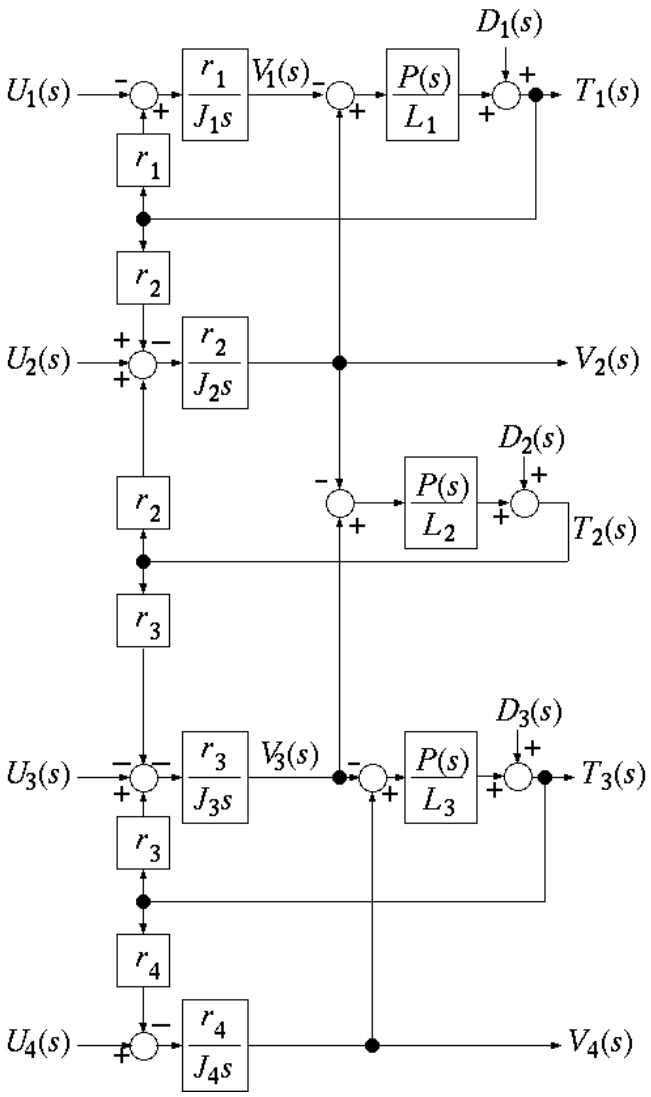
$$T_i(s) = \frac{P(s)}{L_i}\{V_{i+1}(s) - V_i(s)\}$$

$P(s)$ はVoigt（フォークト）モデルに基づくウェブ特性

$$P(s) \triangleq \mathrm{A}(\eta_v + G_v/s)$$

1-2. モデルの変動要因とモデル化の困難さ～モデル化をあきらめよう～

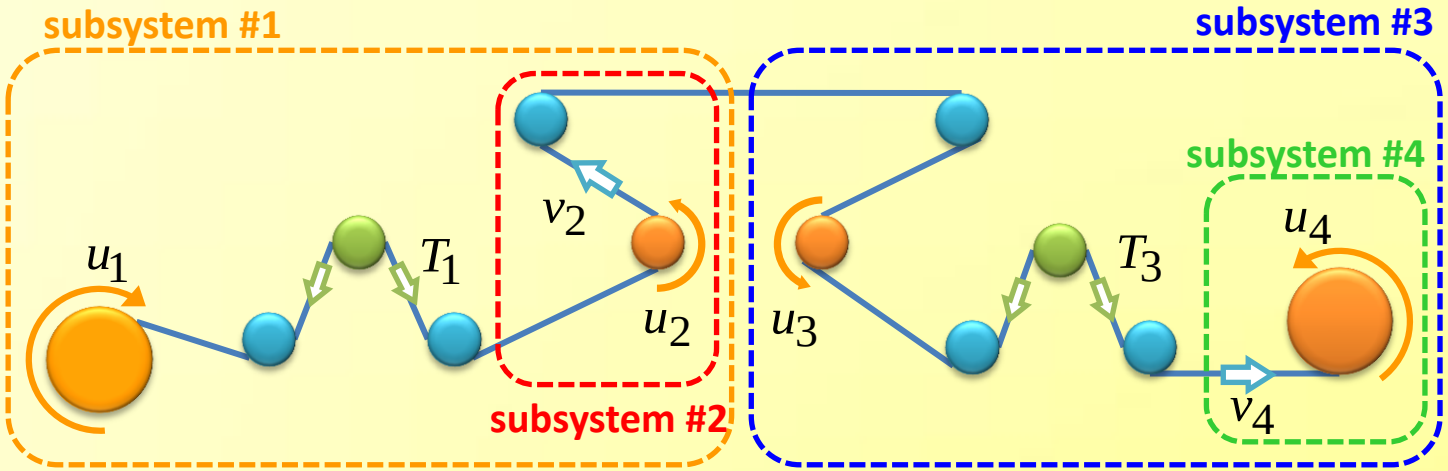
システムのブロック線図



- 強い非線形性を有するためモデル化が困難
- 大規模制御系であるためシステムパラメータの同定が困難
- 分散制御ではサブシステム間の強い相互干渉を無視できない
- 不確かなパラメータ（ローラの摩擦係数やウェブの物性値など）が多数存在
- 多入力多出力系としてモデル化すると，システムの一部を取り換えるような補修ができない。

- 保守や改良に多くの労力が必要
- コントローラのパラメータは現場技術者が試行錯誤で設定

1-3. 重複分割分散制御の構成と困難の克服～すべてSISOで表現しよう～



- 制御量に基づいてシステムを4つのサブシステムに重複分割する
  → 各サブシステムを独立して制御 → 単純な方式で制御可能になる

- 重複分割分散制御法では、強い結合をもつ構成要素を隣接するサブシステムで重複して共有することで物理的に合理性のある分散制御系を構成することができる[1]。

- 重複の無い一般的な分割制御では、各サブシステム間に強い相互干渉力が存在する場合、相互干渉力を低減化するための方策が別途必要となる。

- 重複分割分散制御法は、集中制御系の持つ制御性能に関する長所と、分散制御系の補償器が低次数であるという長所を併せ持つ。

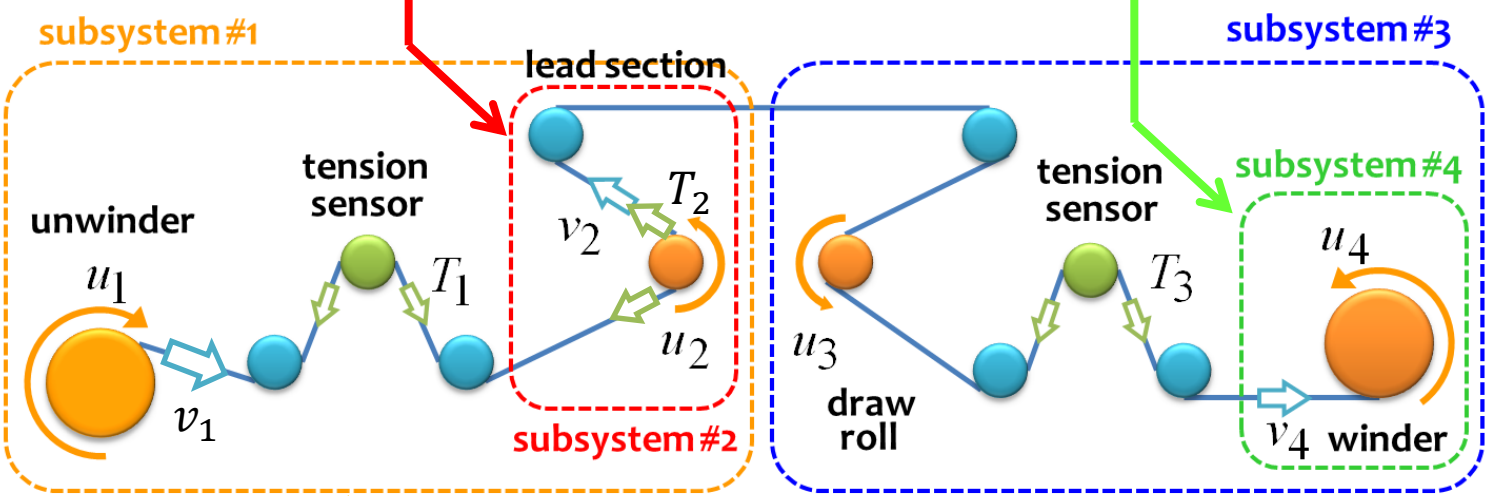
坂本, 田中,“ウェブ張力制御系の重複分割分散制御設計,”
電気学会論文誌, Vol. 118-D, No. 11, pp. 1272-1278 (1997)

1-3. 重複分割分散制御の構成と困難の克服～すべてSISOで表現しよう～

ウェブ速度に関する伝達関数表現

$$V_2(s) = \frac{r_2}{J_2 s}\{U_2(s) + r_2(T_2(s) - T_1(s))\}$$

$$V_4(s) = \frac{r_4}{J_4 s}\{U_4(s) - r_4 T_3(s)\}$$

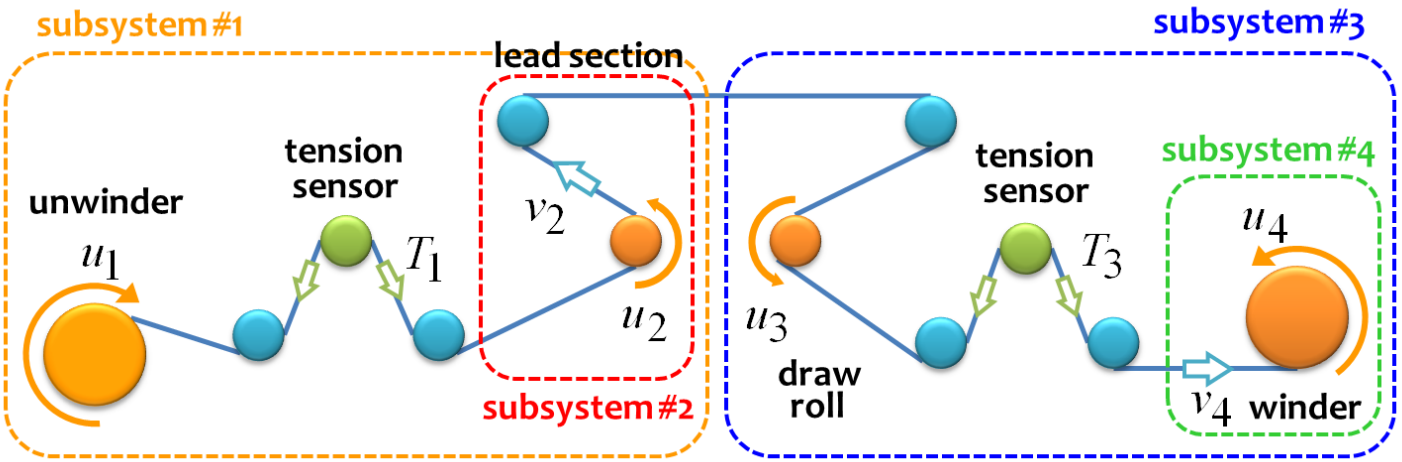


$$V_1(s) = \frac{r_1}{J_1 s}(r_1 T_1(s) - U_1(s))$$

$$V_3(s) = \frac{r_3}{J_3 s}\{r_3(T_3(s) - T_2(s)) - U_3(s)\}$$

1-3. 重複分割分散制御の構成と困難の克服～すべてSISOで表現しよう～

サブシステムの伝達関数（T_iとV_iについてまとめると）



1st と 3rd サブシステム

$$T_i(s) = G_i(s)\tilde{U}_i(s)$$

$$G_i(s) \triangleq \frac{P\,(s)/L_i}{1 + P\,(s)/L_i\left(r_i^2/J_i s + r_{i+1}^2/J_{i+1}s\right)}$$

$$\tilde{U}_i(s) \triangleq \frac{r_i}{J_i}U_i(s) + \frac{r_{i+1}}{J_{i+1}}U_{i+1}(s)$$

2nd と 4th サブシステム

$$V_i(s) = G_i(s)\tilde{U}_i(s) \qquad G_i(s) \triangleq \frac{r_i}{J_i s} \qquad \tilde{U}_i(s) \triangleq U_i(s)$$

1-3. 重複分割分散制御の構成と困難の克服～すべてSISOで表現しよう～

制御入力の算出方法

$$\boldsymbol{u}(t) = \boldsymbol{N}(t)\tilde{\boldsymbol{u}}(t)$$

変換行列を介した入力の算出

$$\tilde{\boldsymbol{u}}(t) = [\tilde{u}_1(t) \quad \tilde{u}_2(t) \quad \tilde{u}_3(t) \quad \tilde{u}_4(t)]$$

各サブシステムの入力ベクトル

$$\boldsymbol{u}(t) = [u_1(t) \quad u_2(t) \quad u_3(t) \quad u_4(t)]$$

各モータへの入力ベクトル

$$\boldsymbol{N}(t) = \begin{bmatrix} \frac{J_1(t)}{r_1(t)} & -\frac{J_1(t)r_2}{J_2 r_1(t)} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{J_2}{r_2} & \frac{J_3 r_4(t)}{J_4(t) r_3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

時変入力変換行列
(ワインダとアンワインダのウェブの量が変化するので注意が必要)

1-3. 重複分割分散制御の構成と困難の克服～すべてSISOで表現しよう～

近似モデル（前述のSISOモデルよりさらに簡略化！）

各サブシステムを一次遅れと零次ホールド要素で表される伝達関数で近似

$$G(s) = \frac{K}{\tau_a s + 1}$$

$$H(s) = \frac{1 - \exp(-\tau_s)}{s}$$

オンライン
で適応同定！
（計算量の圧縮）

$$\mathcal{Z}[H(s)G(s)] = \frac{K\{1 - \exp(-\tau_s/\tau_a)\}z^{-1}}{1 - \exp(-\tau_s/\tau_a)z^{-1}} \triangleq \frac{z^{-1}B(z^{-1})}{A(z^{-1})}$$

τ_s: サンプリング周期

離散モデル

$$y(k) = \frac{z^{-1}B(z^{-1})}{A(z^{-1})}u(k)$$

$$A(z^{-1}) \triangleq 1 - \exp(-\tau_s/\tau_a)z^{-1} \triangleq 1 + a_1 z^{-1}$$

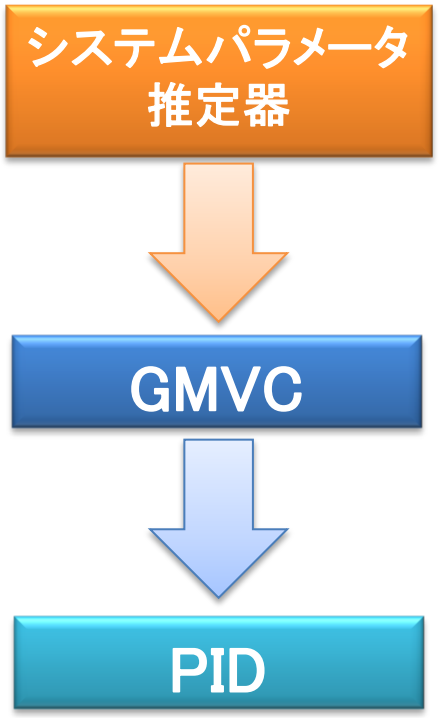
$$B(z^{-1}) \triangleq K\{\exp(\tau_s/\tau_a) - 1\}z^{-1} \triangleq b_0 z^{-1}$$

システムパラメータ$(a_1\ b_0)$が得られれば適切な制御が可能

2. PID制御とGMVC（一般化最小分散制御）

2種類の制御法を組み合わせる手法を利用

1. 大松 繁, 山本 透 編著：セルフチューニングコントロール, コロナ社（1996）
2. 山本 透, 兼田 雅弘：「一般化最小分散制御則に基づくセルフチューニングPID制御器の一設計」, システム制御情報学会論文誌, Vol. 11, No. 1, pp. 1–9,（1998）
3. 山本 透, 満倉 靖恵, 兼田 雅弘：「遺伝的アルゴリズムを用いたPID制御器の一設計」, 計測自動制御学会論文集, Vol. 35, No. 4, pp. 531–537,（1999）



オンラインでシステムパラメータを推定

GMVCで目標値に追従するための適切な入力を算出

GMVCの入力をPIDパラメータに変換して利用

2-1. PID制御と離散時間実装　～PID制御と実装のおさらい～

連続時間系におけるPID制御則

$$u(t) = K_p\left(e(t) + \frac{1}{T_I}\int e(t)\mathrm{d}t + T_D\dot{e}(t)\right) \qquad e(t) = w(t) - y(t)$$

差分近似

$$\dot{e}(t) \fallingdotseq \frac{e(k) - e(k-1)}{\tau_s} \qquad \int e(t)\mathrm{d}t \fallingdotseq \sum_{i=0}^{k} e(i)\tau_s$$

離散時間系におけるPID制御則

$$u(k) = K_p\left(e(k) + \frac{1}{T_I}\sum_{i=0}^{k} e(i)\tau_s + T_D\frac{e(k) - e(k-1)}{\tau_s}\right)$$

2-1. PID制御と離散時間実装　～PID制御と実装のおさらい～

一時刻分の差分

プログラムの実装
はこの形を利用

離散時間系におけるPID制御則

$$u(k)-u(k-1)=K_p\left(1+\frac{\tau_s}{T_I}+\frac{T_D}{\tau_s}\right)e(k)-K_p\left(1+\frac{2T_D}{\tau_s}\right)e(k-1)+K_p\left(\frac{T_D}{\tau_s}\right)e(k-2)$$

$\Delta=1-z^{-1}$を使ってまとめる

$$\Delta u(k)=K_p\left(\Delta+\frac{T_s}{T_I}+\frac{T_D}{T_s}\Delta^2\right)e(k)$$

さらに変形

e.g. $e(k)z^{-1}=e(k-1)$

e.g. $\Delta u(k)=u(k)-u(k-1)$

2-1. PID制御と離散時間実装　～PID制御と実装のおさらい～

$$\Delta u(k) = k_p\left(\Delta + \frac{T_s}{T_I} + \frac{T_D}{T_s}\Delta^2\right)(w(k) - y(k))$$

transform

$$C(z^{-1}) \triangleq c_0 + c_1 z^{-1} + c_2 z^{-2}$$
$$= k_p\left(1 + \frac{T_s}{T_I} + \frac{T_D}{T_s}\right) - k_p\left(1 + \frac{2T_D}{T_s}\right)z^{-1} + \frac{k_p T_D}{T_s}z^{-2}$$

$$\Delta u(k) = C(z^{-1})w(k) - C(z^{-1})y(k)$$

2-2. GMVCとPIDパラメータ ～PIDパラメータ調整はGMVCに任せよう！～

GMVC

評価関数

$$J = E[\phi^2(k + k_m + 1)]$$

$$\phi(k + k_m + 1) \triangleq E[P(z^{-1})y(k + k_m + 1) + \lambda\Delta u(k) - R(z^{-1})w(k + k_m)]$$

応答特性に関する設計多項式　制御入力の調整項　偏差に関する設計多項式

$w(k)$	目標値
λ	制御入力の重みパラメータ
Δ	$\triangleq 1 - z^{-1}$
k_m	$\triangleq 1$, 最短の無駄時間の見積もり

- 計測ノイズが発生する系や無駄時間既知の制御系に適した制御則
- さらに多段階に予測を繰り返す制御則はGPC（一般化予測制御）と呼ばれる

2-2. GMVCとPIDパラメータ ～PIDパラメータ調整はGMVCに任せよう！～

GMVC

制御入力$u(k)$は以下の式で与えられる

$$\Delta u(k) = \frac{R(z^{-1})w(k) - F(z^{-1})y(k)}{E(z^{-1})B(z^{-1}) + \lambda}$$

Diophantine equation

$$P(z^{-1}) = \Delta A(z^{-1})E(z^{-1}) + z^{-(k_m+1)}F(z^{-1})$$

$$E(z^{-1}) \triangleq 1 + e_1 z^{-1}$$
$$F(z^{-1}) \triangleq f_0 + f_1 z^{-1}$$
$$P(z^{-1}) \triangleq 1 + p_1 z^{-1} + p_2 z^{-2}$$

PI制御を考えているのでこの形

$P(z^{-1})$は設計多項式

PID制御を考慮するなら$E(z^{-1})$と$F(z^{-1})$は2次の多項式になる

2-2. GMVCとPIDパラメータ ～PIDパラメータ調整はGMVCに任せよう！～

GMVC

$$P(z^{-1}) \triangleq 1 + p_1 z^{-1} + p_2 z^{-2}$$

$$p_1 \triangleq -2e^{-\frac{\rho}{2\mu}} \cos \frac{\sqrt{4\mu - 1}}{2\mu} \rho$$

$$p_2 \triangleq -e^{-\frac{\rho}{\mu}}$$

$$\rho \triangleq \frac{\tau_s}{\sigma}$$

$$\mu \triangleq 0.25(1 - \delta) + 0.51\delta$$

$\delta = 0$

$$p_1 \triangleq -2e^{-\frac{2\tau_s}{\sigma}}$$

$$p_2 \triangleq e^{-\frac{4\tau_s}{\sigma}}$$

$$\rho \triangleq \frac{\tau_s}{\sigma}$$

$$\mu \triangleq 0.25$$

オーバーシュートの発生を抑える

σで立ち上がりの波形を設計

$$\Delta u(k) = \frac{R(z^{-1})w(k) - F(z^{-1})y(k)}{B(z^{-1})E(z^{-1}) + \lambda}$$

定常応答特性を重視した近似

近似

$$B(z^{-1})E(z^{-1}) \simeq B(1)E(1)$$

$$\upsilon \triangleq B(1)E(1) + \lambda$$

$$\Delta u(k) = \frac{R(z^{-1})w(k) - F(z^{-1})y(k)}{\upsilon}$$

2-2. GMVCとPIDパラメータ ～PIDパラメータ調整はGMVCに任せよう！～

GMVC

$$\Delta u(k) = \frac{R(z^{-1})}{v} w(k) - \frac{F(z^{-1})}{v} y(k)$$

PI controller

$$\Delta u(k) = C(z^{-1}) w(k) - C(z^{-1}) y(k)$$

対応する多項式を等しく設計

$$F(z^{-1})/v = C(z^{-1}) \qquad R(z^{-1}) = F(z^{-1})$$

GMVCの制御入力 → PIパラメータ

$$k_p = -f_1/v$$

$$T_I = -f_1 T_s/(f_0+f_1)$$

$$\begin{cases} f_0 = p_2 + \hat{a}_1(k) + (1 - \hat{a}_1(k))e_1 \\ f_1 = e_1 \hat{a}_1(k) \\ e_1 = p_1 - \hat{a}_1(k) + 1 \\ v = \hat{b}_0(k)(1 + e_1) + \lambda \end{cases}$$

λは入力を抑えるための設計パラメータ

2-2. GMVCとPIDパラメータ ～PIDパラメータ調整はGMVCに任せよう！～

- 一次遅れ系で各サブシステムを近似したので，T_D は算出されない．
- 制御系を構成してしまえば，状況に応じて調整するパラメータはλ のみ．

- 100ステップ前までの応答を利用して算出した推定システムパラメータ$(\hat{a}_1 \ \ \hat{b}_0)$に基づきPIパラメータを自動的に調整する．
- PID制御則によって作動している設備への導入が容易．
- システムの挙動を周波数領域で考慮しやすい．
- 自動的に調整される制御入力の挙動や意味合いをPID制御に慣れた技術者に理解しやすく提示することができる。

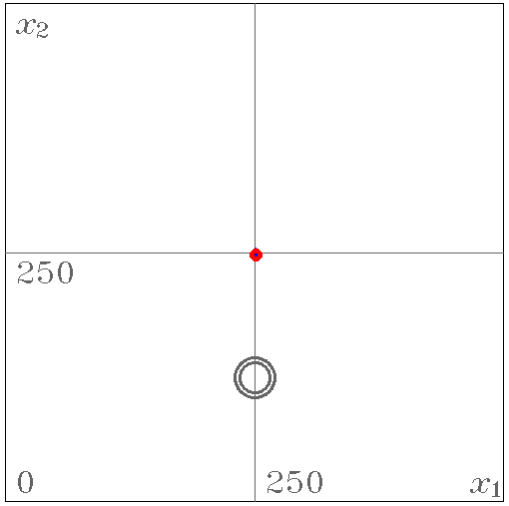
西田 健, 坂本 哲三, ニコラ イバン ジアノカッロ, ``**適応PSOを用いるセルフチューニングPIコントローラによるウェブ搬送系の重複分割分散制御**,'' 電気学会論文誌 D，Vol. 131, No. 12, pp. 1442-1450, 2011.

3. モデルパラメータの適応推定

PSO (particle swarm optimization)

メタヒューリスティクスの一手法であり，確率的な最適解の探索を行う

静的な最適解探索問題の解法として提案された

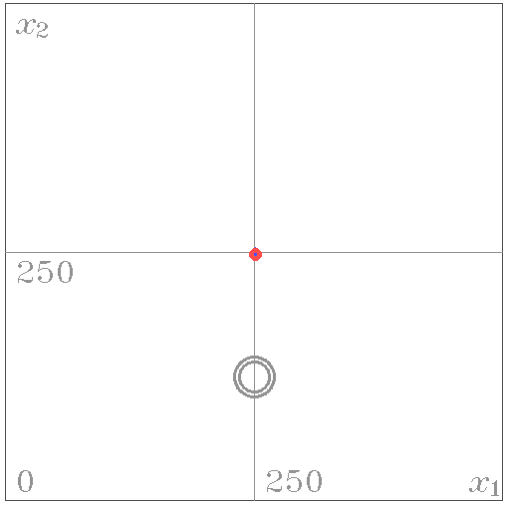


OPSO (online-type particle swarm optimization)

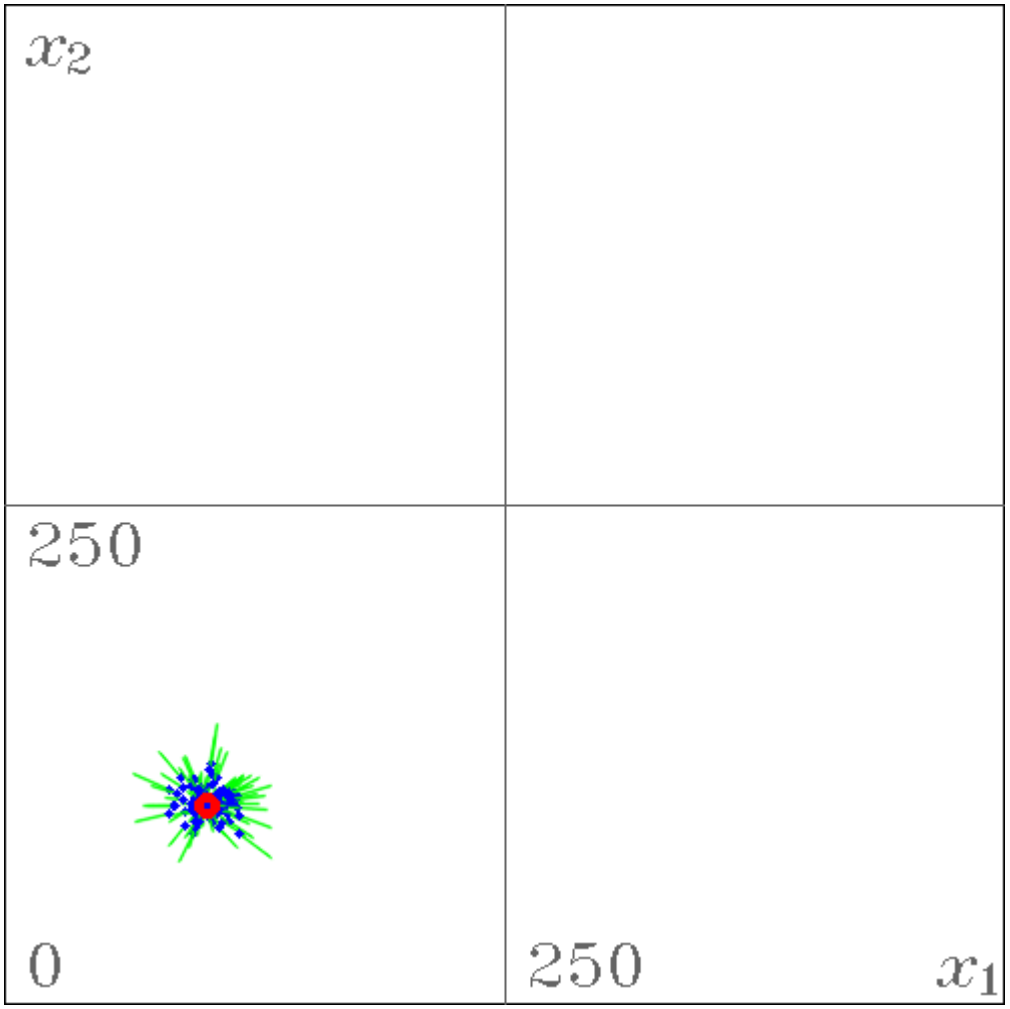
PSOの改良により，動的関数の最適解探索を行うことが可能な手法.

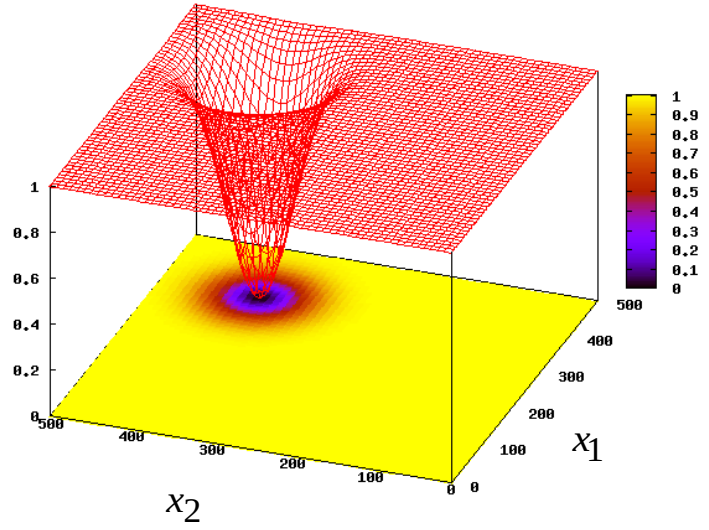
オンラインで実行可能であり，収束性能が良い

西田 健, 坂本 哲三, ``**時変システムのオンライン同定のための適応PSO**,'' 電気学会論文誌 C, Vol. 131, No. 9, pp. 1642-1649, 2011.



3.1 PSO（粒子群最適化）～最適解を探してくれる便利な最適化ツール～





$$f(\boldsymbol{x}) = 1 - \exp\left\{-\frac{1}{2}\left[\frac{(x_1 - b_1)^2}{\sigma_{x_1}^2} + \frac{(x_2 - b_2)^2}{\sigma_{x_2}^2}\right]\right\}$$

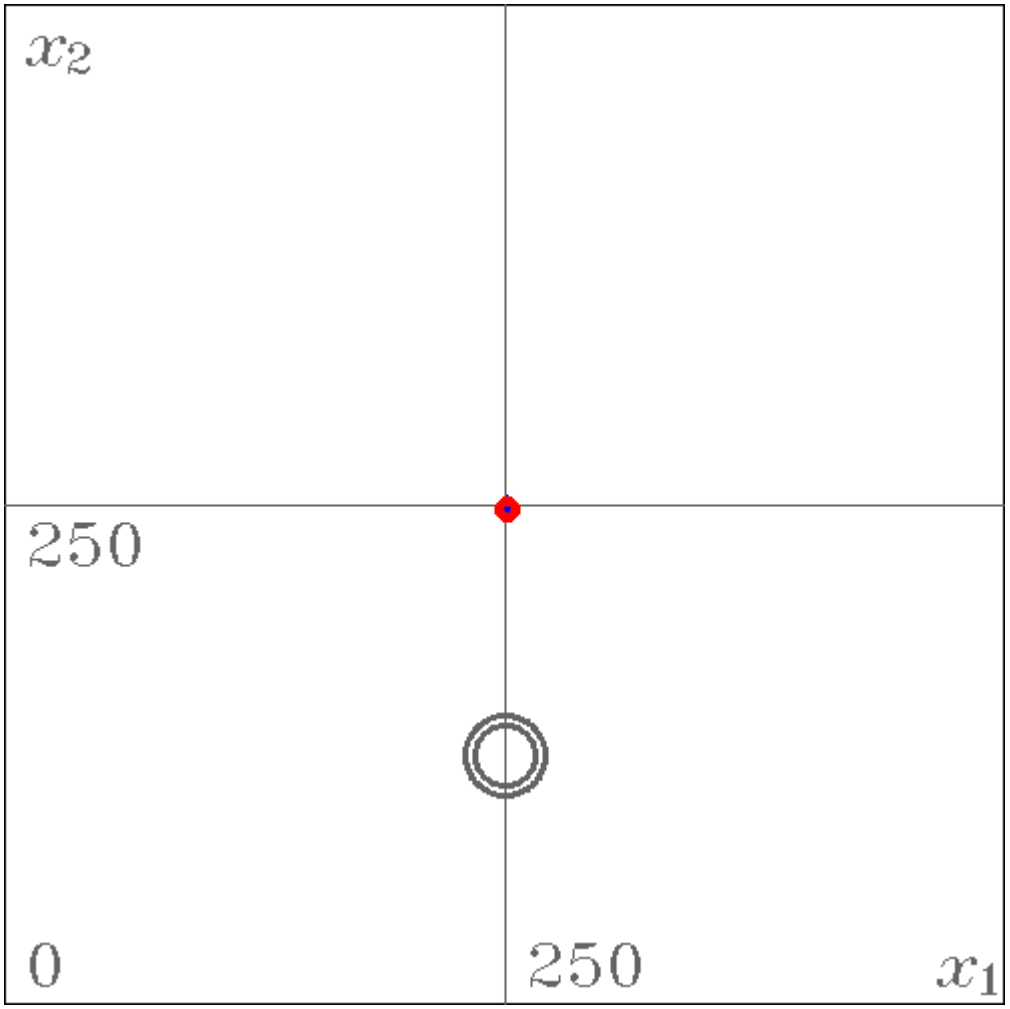
$$(b_1\ b_2) = (375\ \ 375) \qquad (\sigma_1\ \sigma_2) = (40\ \ 40)$$

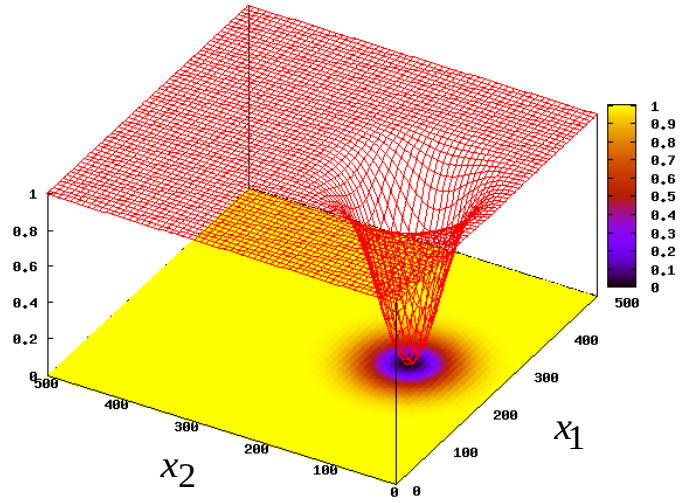
position of gbest

velocity of particle

position of particle

PSOによる時変環境における探索例





$$f_k(\boldsymbol{x}_k) = 1 - \exp\left\{-\frac{1}{2}\left[\frac{(x_1 - b_1 - 125\sin(0.01k))^2}{\sigma_{x_1}^2} + \frac{(x_2 - b_2 - 125\cos(0.01k))^2}{\sigma_{x_2}^2}\right]\right\}$$

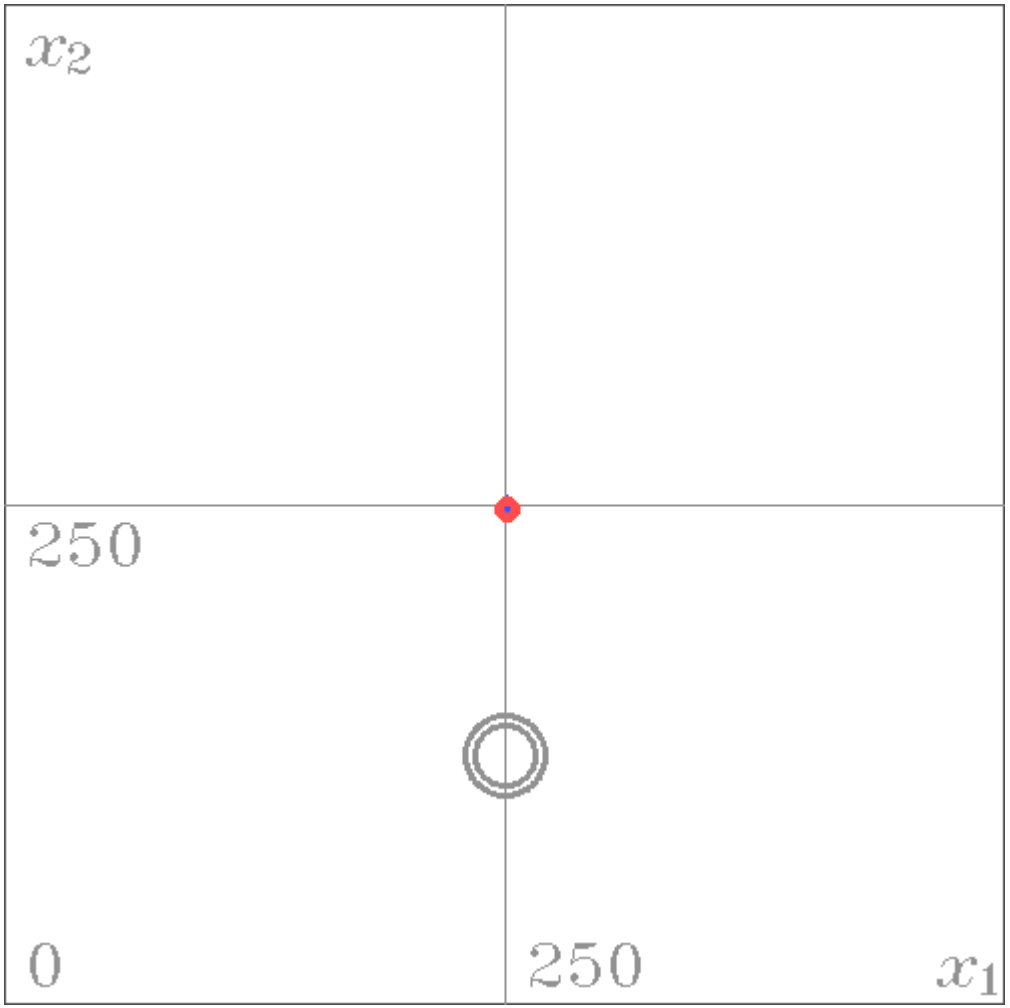
$$(b_1\ b_2) = (375\ \ 375) \qquad (\sigma_1\ \sigma_2) = (40\ \ 40)$$

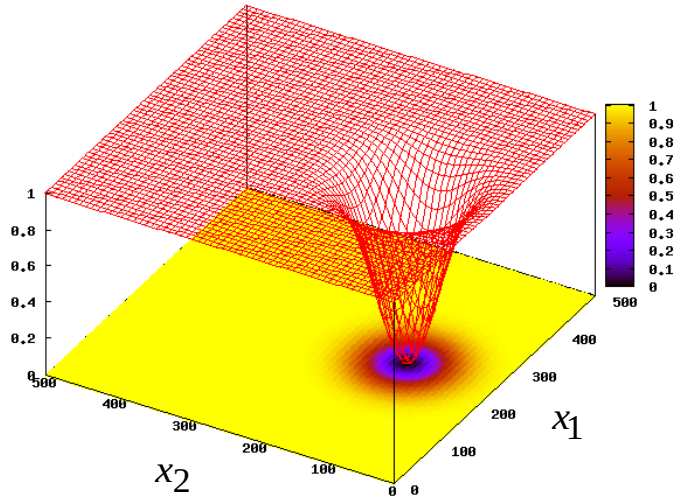
position of target　　velocity of particle

position of gbest　　position of particle

$$c_1 = c_2 = 1.4 \qquad w = 0.8$$

Online-PSOによる時変環境における探索例





$$f_k(\boldsymbol{x}(k)) = 1 - \exp\left\{-\frac{1}{2}\left[\frac{(x_1 - b_1 - 125\sin(0.01k))^2}{\sigma_{x_1}^2} + \frac{(x_2 - b_2 - 125\cos(0.01k))^2}{\sigma_{x_2}^2}\right]\right\}$$

$$(b_1\ b_2) = (375\ \ 375) \qquad (\sigma_1\ \sigma_2) = (40\ \ 40)$$

target position

solution

velocity of particle

position of particle

$$c_1 = c_2 = 1.4 \qquad w = 0.8$$

3.2 時間変化するパラメータをOPSOで適応推定 ～最新PSOアルゴリズム～

粒子の評価

評価ステップ数

出力

推定出力

$$f_k(\boldsymbol{x}_m(k)) = \sum_{i=0}^{I_e} \| y(k-i) - \boldsymbol{\psi}^T(k-i)\boldsymbol{\theta}_m(k) \|$$

第m粒子の推定

出力と入力の組み合せ

$\boldsymbol{\psi}^T(k-i) \triangleq (-y(k-i-1) \quad u(k-i-1))$

I_eの値を大きく設定すると
計測ノイズの影響を低減化できる

trade off

- 計算量が増加する
- 推定の無駄時間が増加する

- 実験では $I_e = 100$ に設定
- 実験機のサンプリング時間は $T_s = 10$ [ms]
  現在までの1 秒間の応答を利用して適応的な推定を行う
- ! 推定にむだ時間や遅れが生ずる

3.2 時間変化するパラメータをOPSOで適応推定 ～最新PSOアルゴリズム～

OPSO algorithm

定式化

最適化問題:

$$\min_{\boldsymbol{x}} f_k(\boldsymbol{x}) \geq 0$$

$f_k: \mathbb{R}^2 \to \mathbb{R}$: 時変評価関数

$\boldsymbol{x}_m(k) \triangleq (a_{1m}(k)\ \ b_{0m}(k))^T$	探索空間内の粒子の位置
$\boldsymbol{v}_m(k) \in \mathbb{R}^2$	粒子の速度
$m = 1,2,\cdots,M$	粒子番号

1 前時刻の各粒子の最良解$\hat{\boldsymbol{x}}_m(k-1)$を現時刻の評価関数$f_k(\cdot)$で再評価し，それらの最小値を求める:

$$\tilde{\boldsymbol{x}}^g(k) = \arg\min\{f_k(\hat{\boldsymbol{x}}_m(k-1))\}$$

3.2 時間変化するパラメータをOPSOで適応推定 ～最新PSOアルゴリズム～

OPSO algorithm

2 各粒子の位置と速度を更新

$$\boldsymbol{v}_m(k) = \omega \boldsymbol{v}_m(k-1) + c_1 r_1 \{\tilde{\boldsymbol{x}}^g(k) - \boldsymbol{x}_m(k-1)\} + c_2 r_2 \{\hat{\boldsymbol{x}}_m(k-1) - \boldsymbol{x}_m(k-1)\}$$

$$\boldsymbol{x}_m(k) = \boldsymbol{x}_m(k-1) + \boldsymbol{v}_m(k)$$

3 各粒子の最良値を更新

$$\hat{\boldsymbol{x}}_m(k) = \begin{cases} \boldsymbol{x}_m(k), & \text{if } f_k(\boldsymbol{x}_m(k)) < f_k(\hat{\boldsymbol{x}}_m(k-1)) \\ \hat{\boldsymbol{x}}_m(k-1), & \text{otherwise} \end{cases}$$

4 最適解（時刻kにおけるOPSO の推定）を更新

$$\hat{\boldsymbol{x}}^g(k) = \arg\min\{f_k(\hat{\boldsymbol{x}}_m(k))\} \triangleq (\hat{a}_1(k) \;\; \hat{b}_0(k))^T$$

従来型 PSO algorithm

Given : w, c_1, c_2

Initialize: $\boldsymbol{x}_m(0), \boldsymbol{v}_m(0)$ set at random

loop

2

$$\boldsymbol{v}_m(k) = w\boldsymbol{v}_m(k-1) + c_1 r_1(\hat{\boldsymbol{x}}_m(k) - \boldsymbol{x}_m(k-1)) + c_2 r_2(\hat{\boldsymbol{x}}^g(k) - \boldsymbol{x}_m(k-1))$$

$$\boldsymbol{x}_m(k) = \boldsymbol{x}_m(k-1) + \boldsymbol{v}_m(k)$$

3

for $m := 1$ to M do

if $f_k(\boldsymbol{x}_m(k)) < f_k(\hat{\boldsymbol{x}}_m(k))$ then

$\hat{\boldsymbol{x}}_m(k) := \boldsymbol{x}_m(k)$

4

if $f_k(\boldsymbol{x}_m(k)) < f_k(\hat{\boldsymbol{x}}^g(k))$ then

$\hat{\boldsymbol{x}}^g(k) := \boldsymbol{x}_m(k)$

end if

end if

end for

$k := k + 1$

end loop

OPSO algorithm

Given : w, c_1, c_2

Initialize: $\boldsymbol{x}_m(0), \boldsymbol{v}_m(0)$ set at random

loop

for $m := 1$ to M do

$f_k(\hat{\boldsymbol{x}}_m(k)) := f_k(\hat{\boldsymbol{x}}_m(k-1))$

end for

1 for $m := 1$ to M do

if $f_k(\hat{\boldsymbol{x}}_m(k)) < f_k(\hat{\boldsymbol{x}}^g(k))$ then

$\tilde{\boldsymbol{x}}^g(k) := \hat{\boldsymbol{x}}_m(k)$

end if

end for

2 3 4 conventional PSO

$k := k + 1$

end loop

この部分が従来型PSOと異なる

OPSO の特徴

Good 1 従来型と設定パラメータの数は同じ

Good 2 計算手順がシンプル

ε PSO algorithm

Given : w, c_1, c_2

Initialize: $\boldsymbol{x}_0^{(m)}, \boldsymbol{v}_0^{(m)}$ set at random

loop

$f_k(\hat{\boldsymbol{x}}_k^g) := f_{k-1}(\hat{\boldsymbol{x}}_{k-1}^g) + \varepsilon_g$

for $m := 1$ to M do

$f_k(\hat{\boldsymbol{x}}_k^{(m)}) := f_{k-1}(\hat{\boldsymbol{x}}_{k-1}^{(m)}) + \varepsilon_p$

end for

2

3

4

conventional PSO

$k := k + 1$

end loop

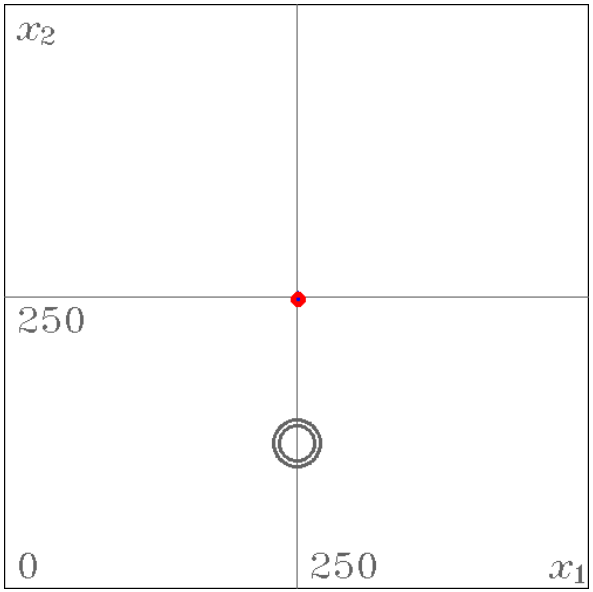
すべての粒子の最良値と全体の最良値に
摂動のための微小な値を加える

advantage　計算量の増加がほとんどない

disadvantage　調整パラメータが増える $\left(\varepsilon_g, \varepsilon_p, \Sigma\right)$

各種PSOによる時変環境における探索例

conventional PSO

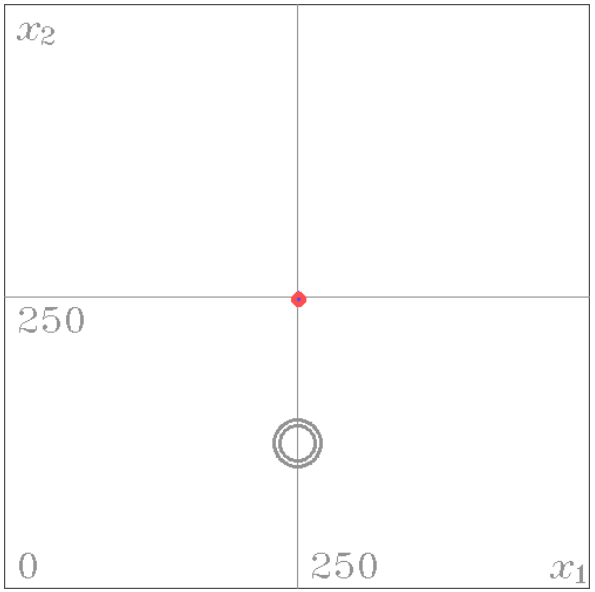


advantage

disadvantage

時変環境における利用ができない

online type PSO



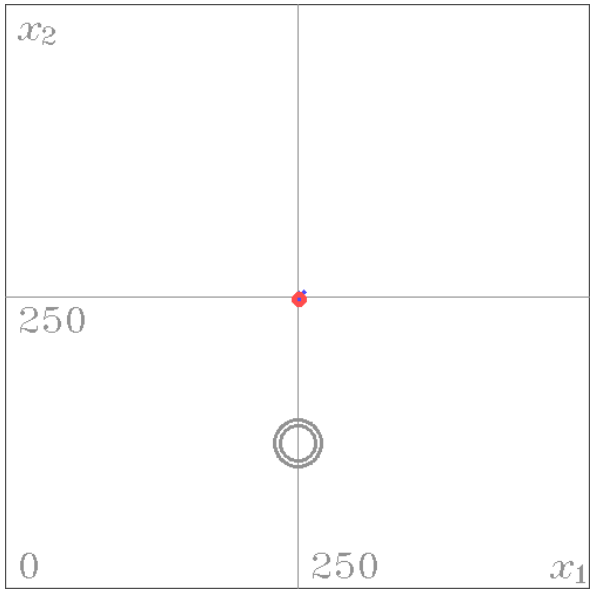
advantage

調整パラメータは増えない

disadvantage

計算量の増加がε PSOより大きい

ε PSO



advantage

計算量の増加がほとんどない

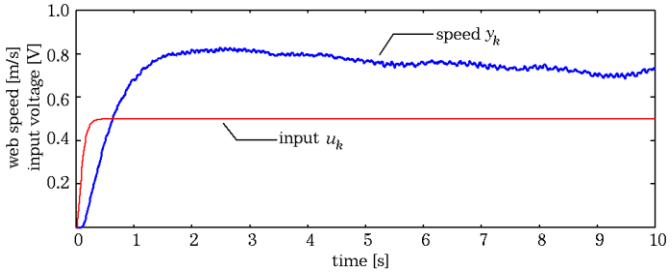
disadvantage

調整パラメータが増える $\left(\varepsilon_g, \varepsilon_p, \Sigma\right)$

PSOを利用するセルフチューニングGMVC-PI制御系の処理手順

1 サブシステムの入出力の値を獲得する

$$\boldsymbol{z}_k = (y_{k-1}\ y_{k-2}\ u_{k-1}\ u_{k-2}\ u_{k-3})^T$$



2 PSOによりプラントのパラメータを探索（推定）する

プラントの伝達関数

$$y_k = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} u_k$$

プラントのパラメータ

$$\boldsymbol{x}_k = (a_1\ a_2\ b_0\ b_1\ b_2)^T$$

PSOの評価関数

$$f_k\left(\boldsymbol{x}_{ik}^{(m)}\right) = \frac{1}{I}\sum_{j=0}^{I}\left\| y_{k-j} - \hat{y}_{k-j}^{(m)} \right\| = \frac{1}{I}\sum_{j=0}^{I}\left\| y_{k-j} - \boldsymbol{x}_{k-j}^{(m)T}\boldsymbol{z}_{k-j} \right\|$$

3 探索されたパラメータからGMVC-PIDの手法に基づいてPIDゲインを決定

$$\hat{\boldsymbol{x}}_k = (\hat{a}_1\ \hat{a}_2\ \hat{b}_0\ \hat{b}_1\ \hat{b}_2)^T \rightarrow \{K_P\ K_I\ K_D\}$$

4. 実験機の構成と実装ノウハウ

計算量について

計算量低減化 ⇔ トレードオフ ⇔ 高い精度のモデルに基づく制御・制御性能向上

- 低次モデルを採用し，推定問題の探索空間を縮小
- メタヒューリスティクスアルゴリズムを改良し，高効率なパラメータ推定を実現

制御手法のC言語実装について

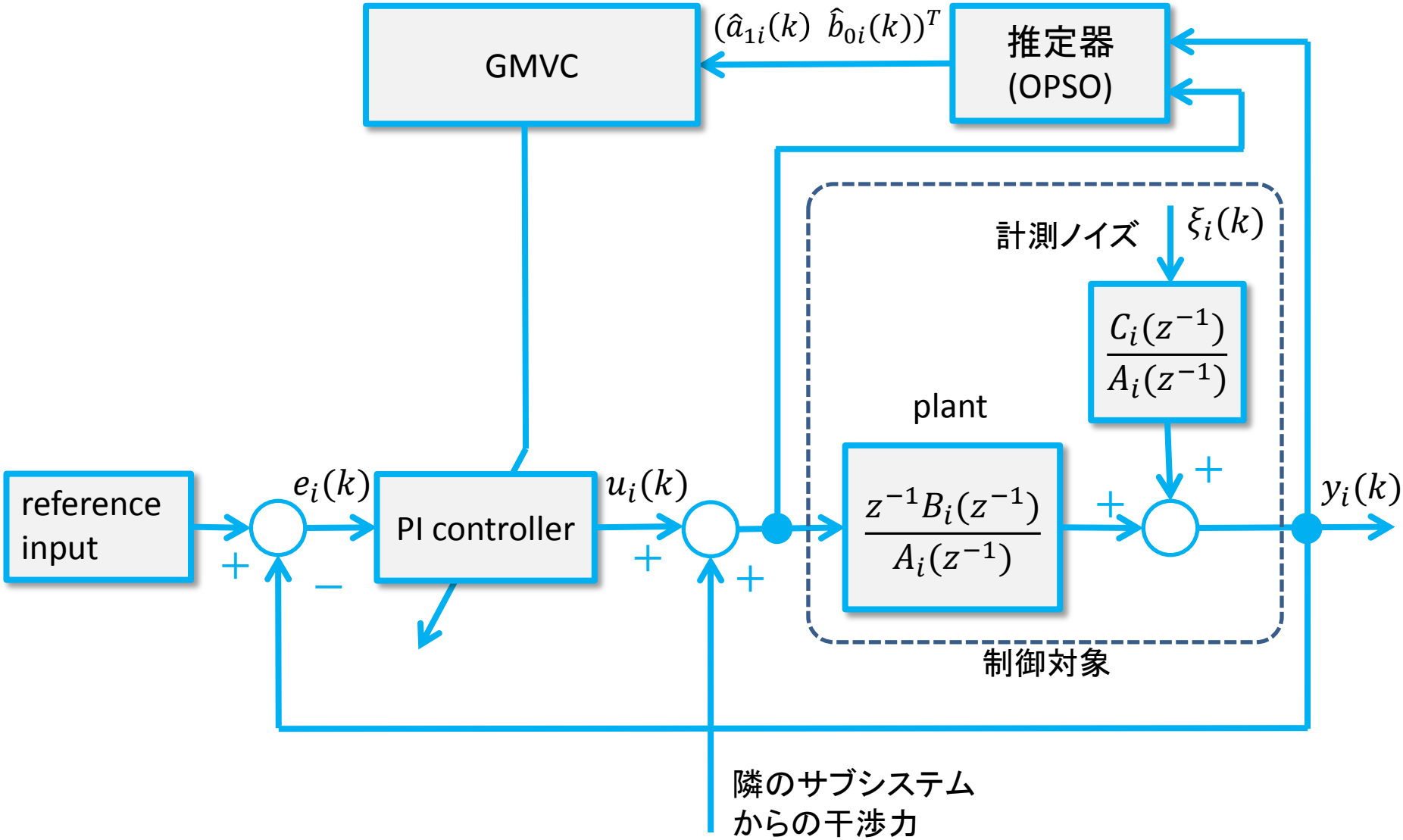
- リアルタイムOSを利用してもサンプリング周期・制御周期が厳密に一定にならない。
- 連続系で構成した制御アルゴリズムを離散系に近似する必要がある。
- 内部クロックでサンプリング周期を計測し，毎回τ_dを変更しながら制御アルゴリズムを適用する。→市販のC言語インタプリタはサンプリング周期の揺らぎに対応できない。

実験機について

- ウェブのすべり，横ずれ，静電気の発生などが，実際には生ずることがある。
- 気温や湿度によってローラ軸受けの潤滑油の特性が大きく変動する。
- ACサーボモータの利用により，センサの計測信号に大きなノイズが混入する。
- 計測ノイズ対策のノイズフィルタによって，制御に遅延を生じさせると同時に，各サブシステム間の相互干渉を増幅し，ウェブシステム全体が不安定化する場合がある。
- 入力飽和を考慮した目標値の設定は，経験に頼る必要がある。

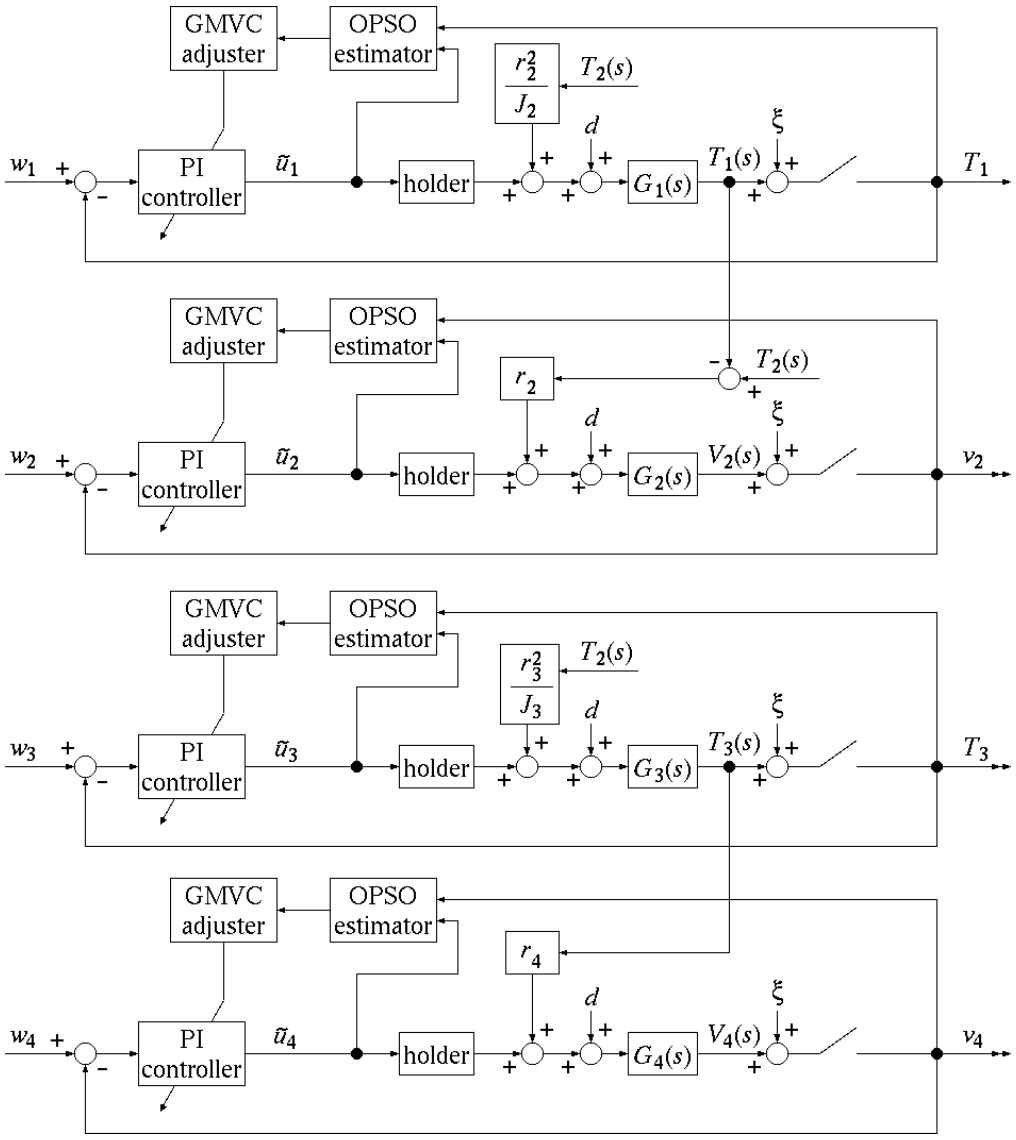
4-1. 実験に使ったウェブ搬送機

1つのサブシステムに対する制御系



4-1. 実験に使ったウェブ搬送機

4つのサブシステムに対する制御系



各サブシステムに同じ制御機構を構成

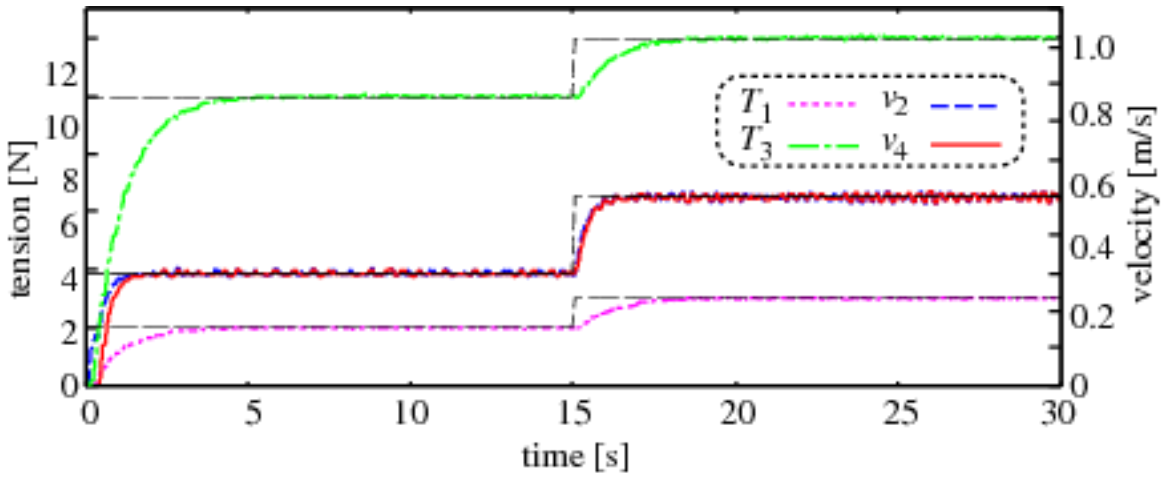
各パラメータの設定

OSPOの粒子数	$M = 100$
評価ステップ数	$I = 100$
GMVC-PID の設計パラメータ	$\lambda = 10$

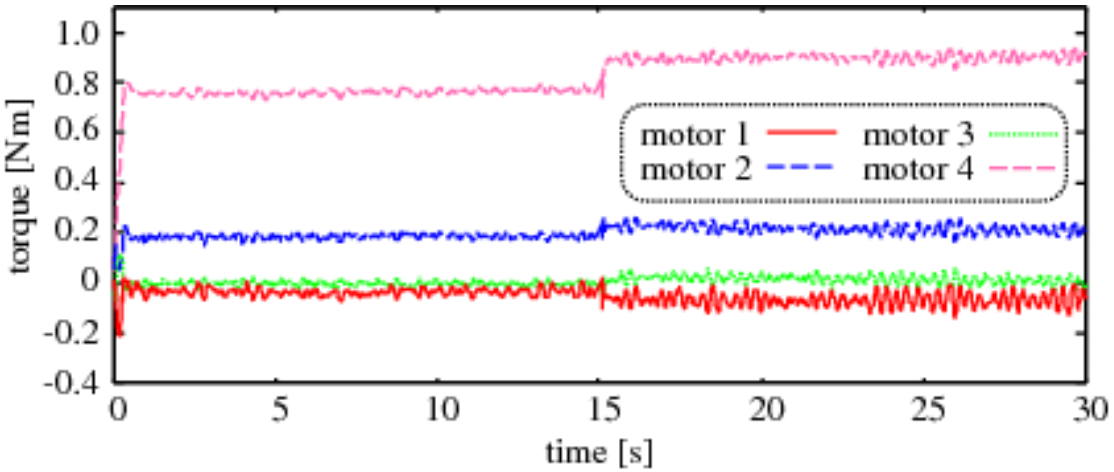
実験に利用した目標値

$0 \leq t \leq 15$ [m/s]	$15 < t \leq 30$ [m/s]
$w_1 = 0.3$ [m/s]	$w_1 = 0.5$ [m/s]
$w_2 = 2$ [N]	$w_2 = 3$ [N]
$w_3 = 10$ [N]	$w_3 = 12$ [N]
$w_4 = 0.3$ [m/s]	$w_4 = 0.5$ [m/s]

4-2. 大きなノイズが発生するセンサと安いPCで高精度の制御を達成する！



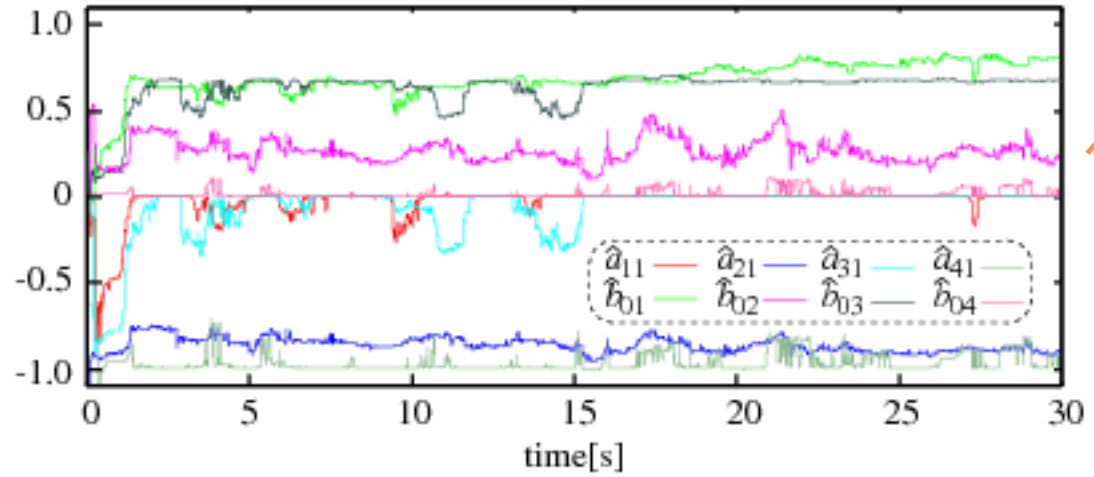
出力値の時間推移



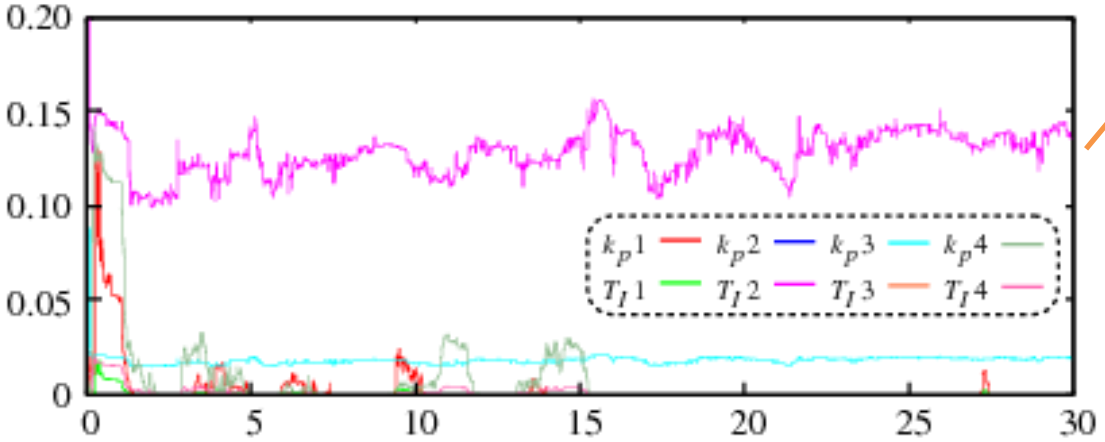
モータへの入力トルクの時間推移

- ✓ オーバーシュートの発生なし
- ✓ 定常偏差なし
- ✓ 入力の変化に対する適応

4-2. 大きなノイズが発生するセンサと安いPCで高精度の制御を達成する！



システムパラメータの
推定値の時間推移



算出されたPIパラメータ
の時間推移

- ✓ システムパラメータの適切な推定が行われている
- ✓ 推定に基づいて適切なPIパラメータが算出されている

4-2. 大きなノイズが発生するセンサと安いPCで高精度の制御を達成する！

手動チューニングしたPI制御系との性能比較

1

ZN法やCHR法により算出したPIパラメータによる制御実験を行ったが，制御開始直後にウェブが破断し制御不能となった.

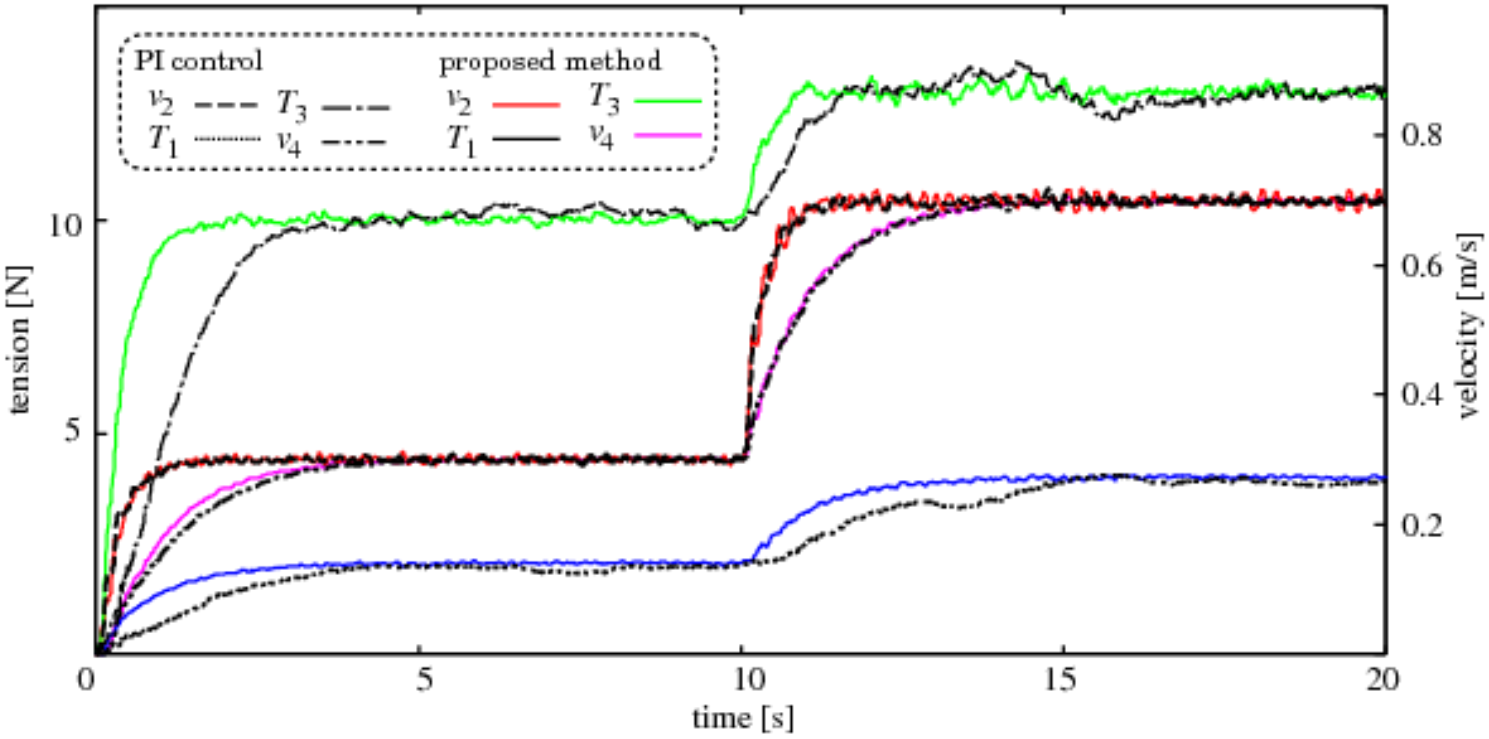
2

前述のPIパラメータ決定法では，速度制御系である2nd および 4th サブシステムのk_p の値が比較的大きく設定される傾向がある.
そこで，まず張力制御系である1st および3rd サブシステムのパラメータをCHR法により調整し，その後，k_p の値を徐々に増しながら複数回の試行錯誤を行い2nd と4th サブシステムのPIパラメータを決定した.

No. of subsystem	1	2	3	4
k_p	0.01	0.15	0.30	0.15
T_I	0.013	0.020	0.175	0.020

4-2. 大きなノイズが発生するセンサと安いPCで高精度の制御を達成する！

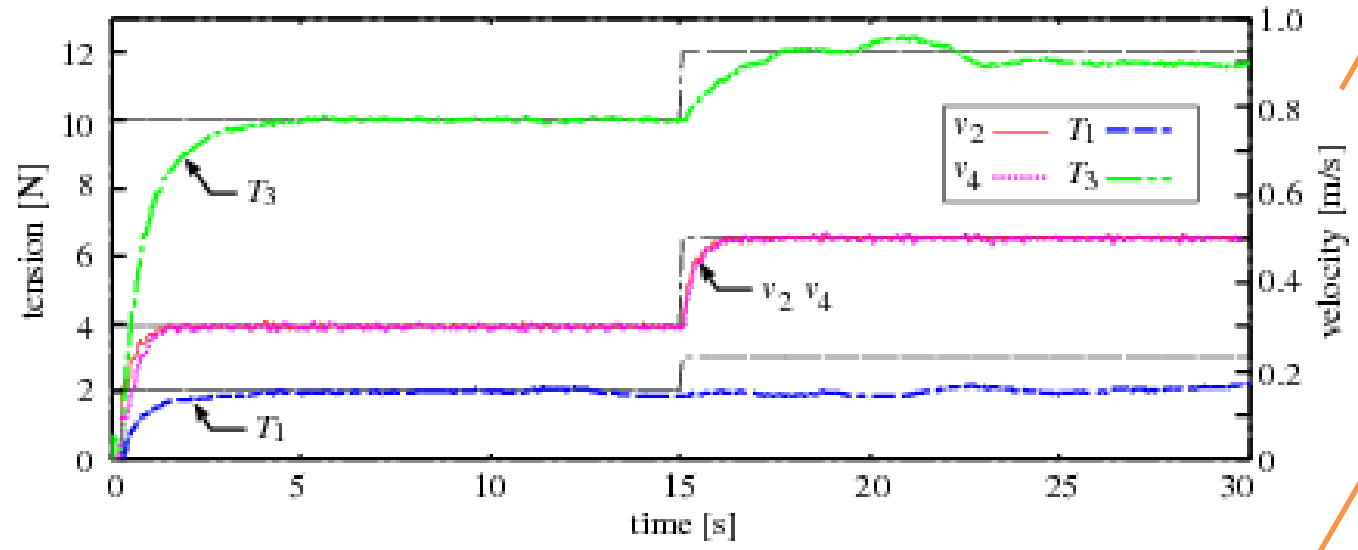
手動チューニングしたPI制御系との性能比較



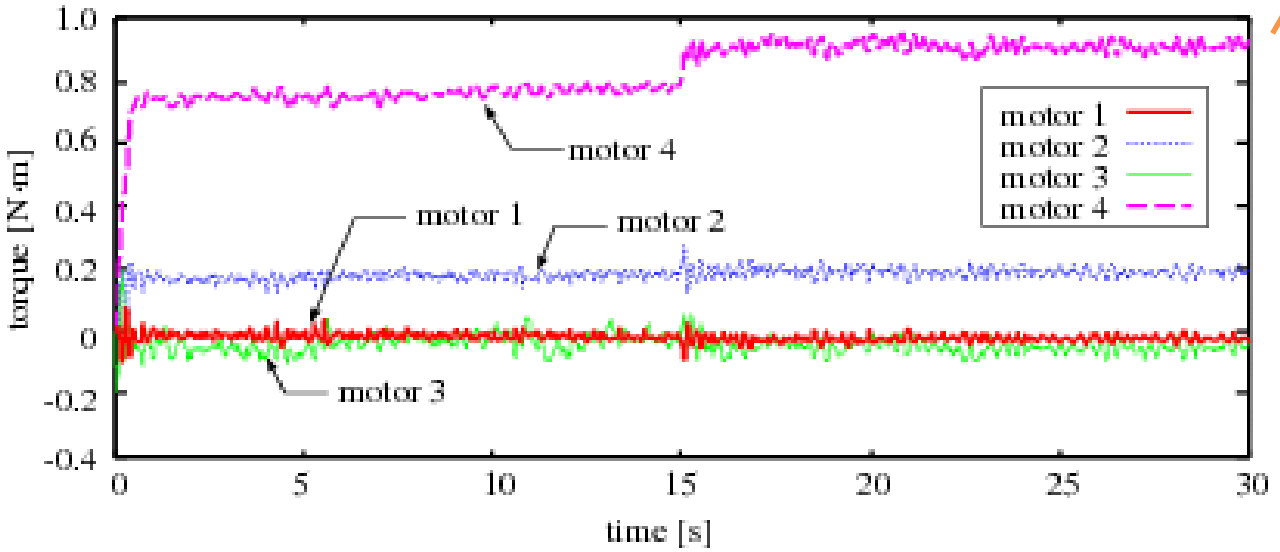
苦労して試行錯誤的に設計したPI制御器よりも，セルフチューニングPI制御器の方が，高い制御性能を有していることが分かる．

4-2. 大きなノイズが発生するセンサと安いPCで高精度の制御を達成する！

RLS（逐次最小二乗法）法によりパラメータ推定を行った結果



出力の時間推移

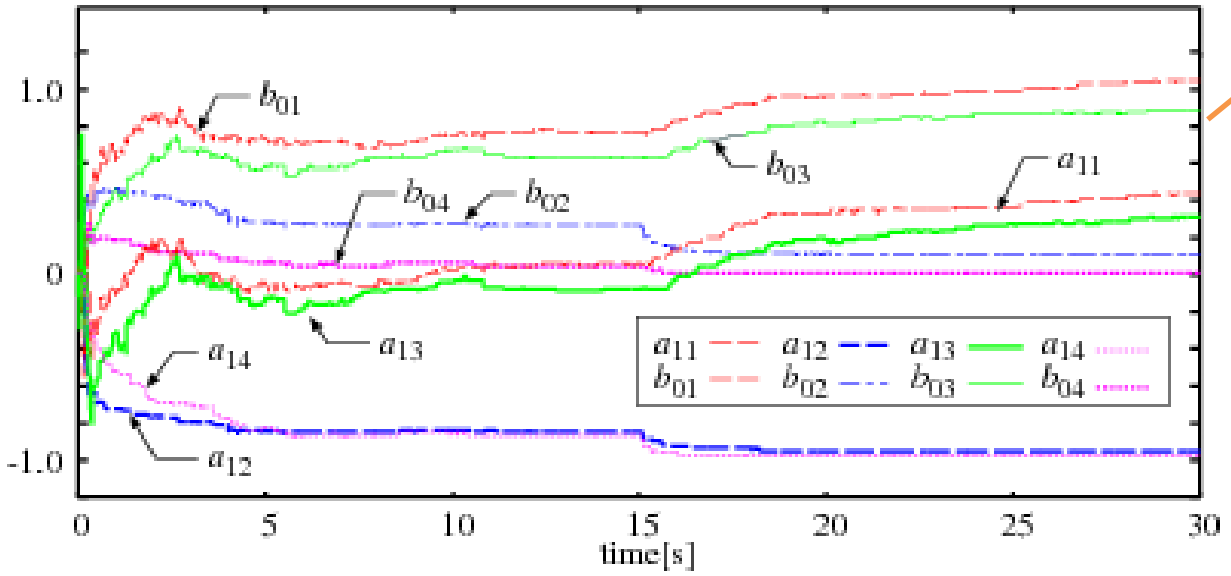


入力の時間推移

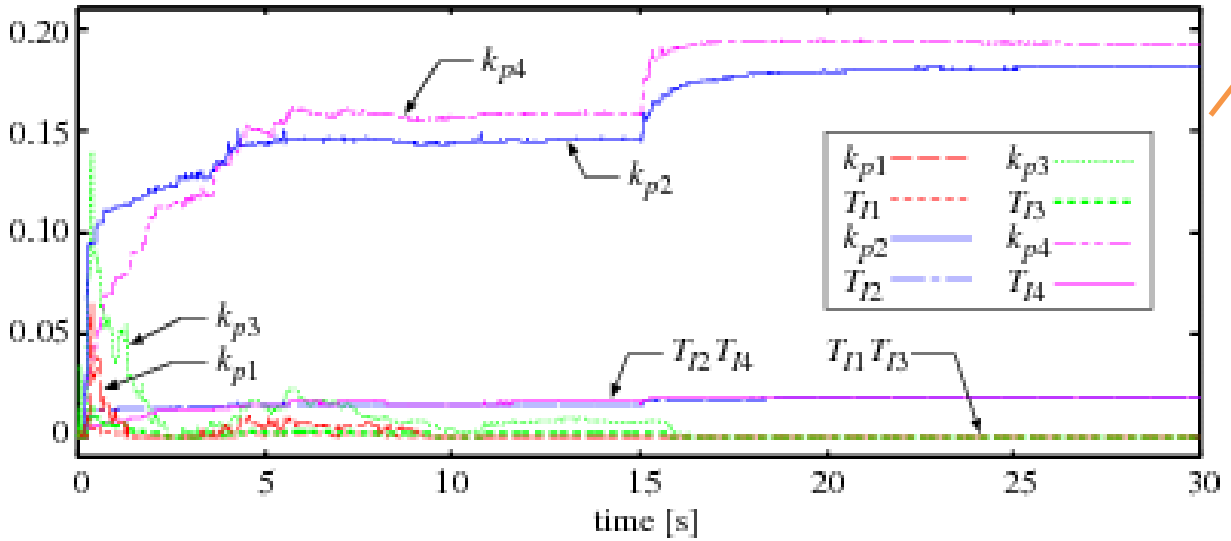
- ✓ オーバーシュート
- ✓ 定常偏差
- ✓ 過渡応答特性
- ✓ 無駄時間
- ✗ 適応性能

4-2. 大きなノイズが発生するセンサと安いPCで高精度の制御を達成する！

RLS（逐次最小二乗法）法によりパラメータ推定を行った結果



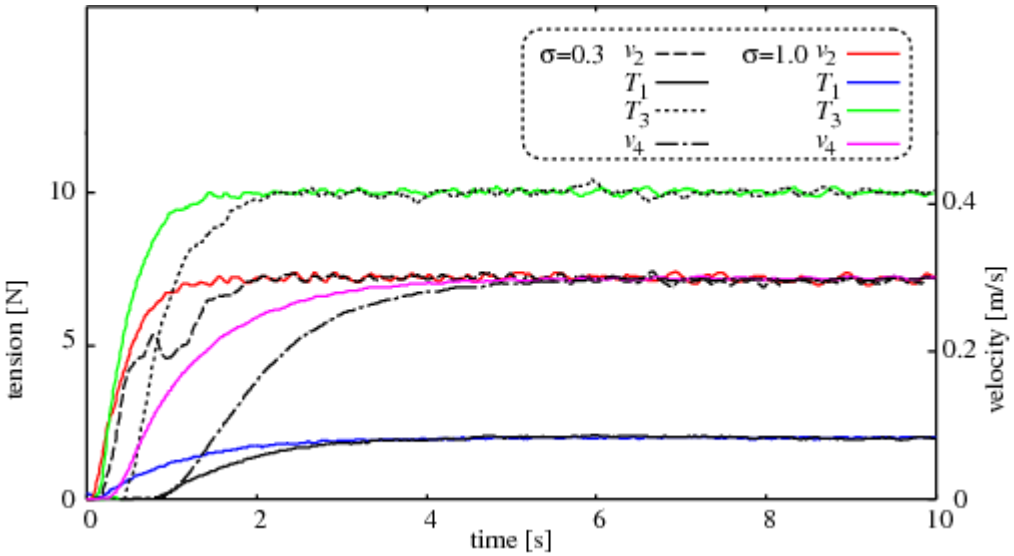
推定パラメータの
時間推移

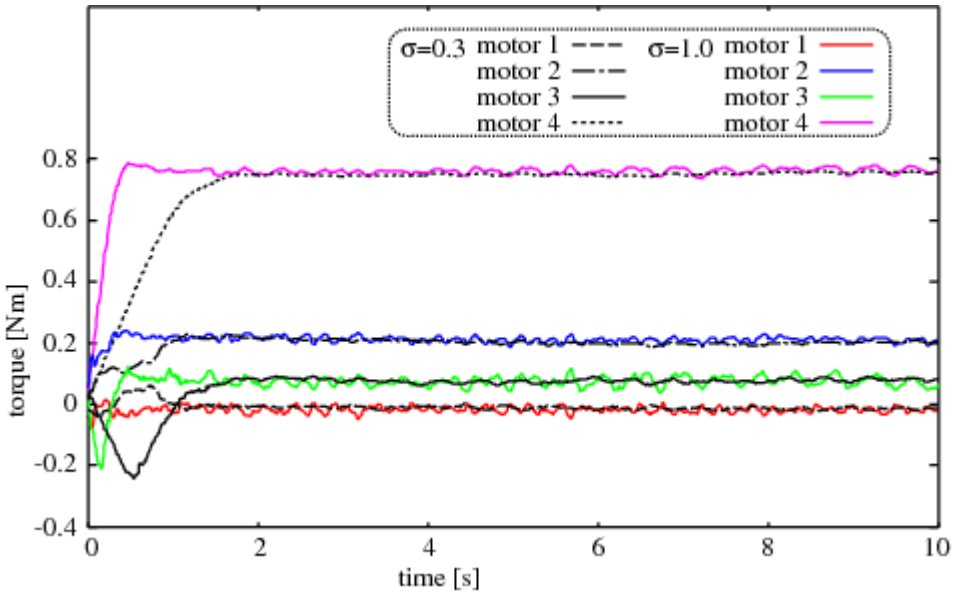


PIパラメータの
時間推移

4-2. 大きなノイズが発生するセンサと安いPCで高精度の制御を達成する！

σの値を変化させた場合の制御性能の変化





初期応答

$\sigma = 1.0$: 速い

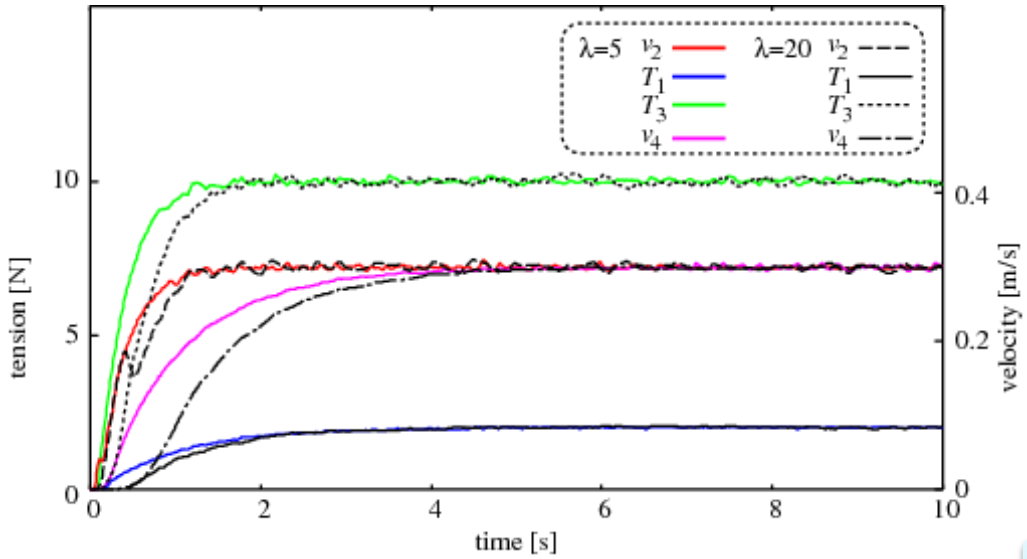
入力の振動

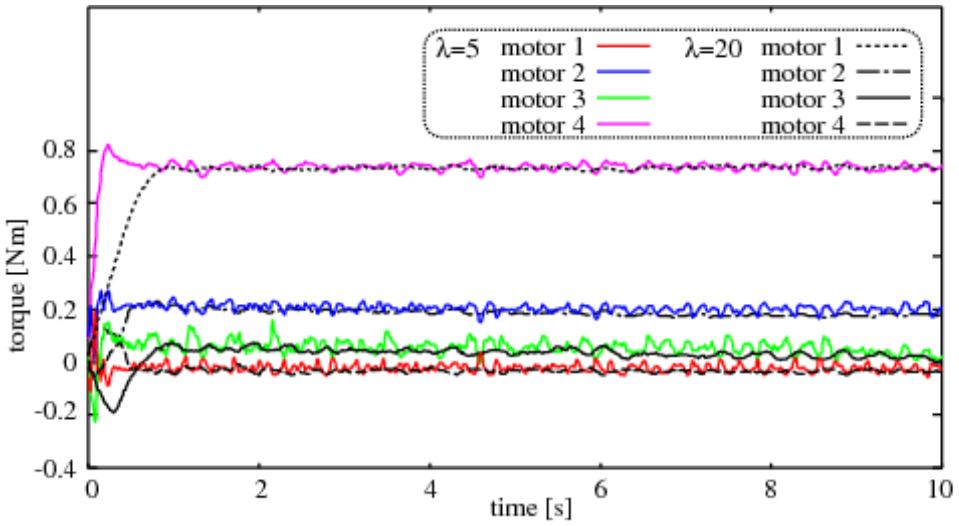
$\sigma = 0.3$: $u(k)$の変化量が制限されるので小さい

入力の振動や初期応答を考慮して経験的に$1.0 \geq \sigma$と設定すると良い。

4-2. 大きなノイズが発生するセンサと安いPCで高精度の制御を達成する！

λの値を変化させた場合の制御性能の変化





初期応答

$\lambda = 5$: 速い

入力の振動

$\lambda = 20$: $u(k)$の変化量が制限されるので小さい

入力の振動や初期応答を考慮して
経験的に$3 \leq \lambda \leq 30$で調整すると良い。

5. おわりに

セルフチューニングGMVC-PI制御系には次のような特徴がある:

- Good 1: 提案手法は一般的な性能の計算機でオンライン実行可能
- Good 2: 設計パラメータの物理的な意味が明確であるので調整が容易
- Good 3: システム全体で設計パラメータは1つのみ（λだけ）である
- Good 4: システムパラメータをオンラインで適応推定するので
  精度の高い対象のシステムモデルが必要無い