Hosoda Lab.
Adaptive Robotics Group, Osaka University

ロボメカ・チュートリアル・ 2022年度版

ロボット制御・教育学習再考

細田　耕【著】
株式会社アールティ【協力】

多自由度化するロボットの制御を実践することを念頭に置き，**解析的計算の詳述や数値計算の工夫**を踏まえながら，ロボット制御技術を整理して説明しています.
また，現状のモータ**周辺装置の実装レベル**を踏まえながら，**必要なところまで読めば**実践できるように構成を工夫しています.

ロボットを学ぶ教科書

大阪大学
OSAKA UNIVERSITY
OPEN 2021
Hosoda Lab.
Adaptive Robotics Group, Osaka University

しっかりした理論的教科書は，
ほとんどが1990年ころに書かれている
動力学計算が盛んに研究された時代
研究テーマがゴールで書かれた教科書が多い
（Craig，Paul，吉川，川崎，大須賀，内山・中村，有本）

その結果，参考書としてはあげられているが
教科書として採用されている本はほとんどない

シラバスで，教科書指定されているのは（川崎）くらい

大阪大学
OSAKA UNIVERSITY
OPEN 2021
Hosoda Lab.
Adaptive Robotics Group, Osaka University

「ロボット」を学ぶ教科書

問題点：

周辺計算に関する意識が薄い
モジュールプログラムの意識なし
学習のゴールは動力学

教科書の「しかけ」

大阪大学 OSAKA UNIVERSITY　OPEN 2021
Hosoda Lab. Adaptive Robotics Group, Osaka University

初学者にやさしい教科書．でもしっかり理論は押さえる

コンピュータ科学，IoTの時代に即した知識を提供する

退屈な理論ではなく，数学･物理学が実践へとつながる

教科書の「しかけ」

初学者にやさしい教科書. でもしっかり理論は押さえる

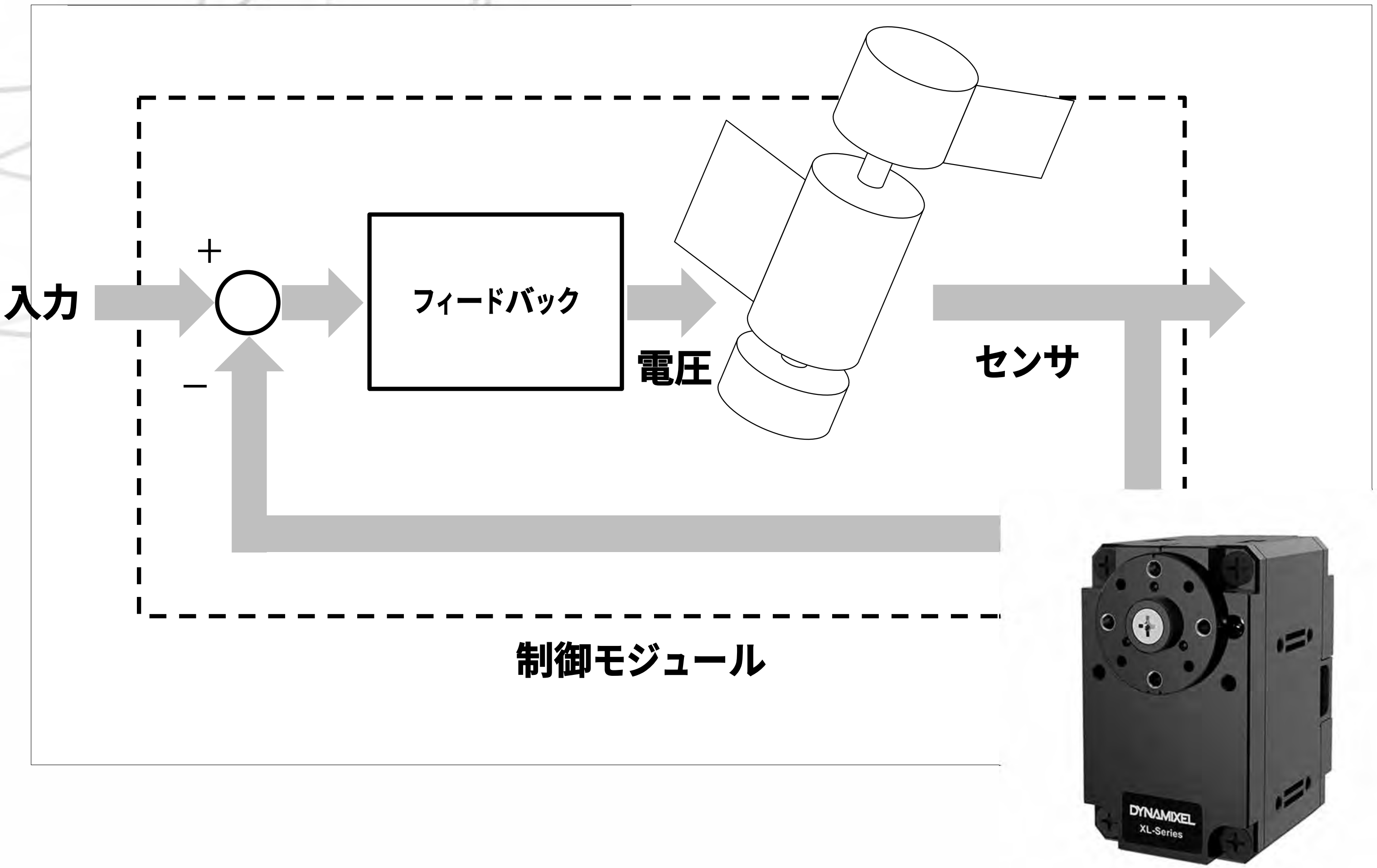
コンピュータ科学, IoTの時代に即した知識を提供する

退屈な理論ではなく, 数学・物理学が実践へとつながる

ロボット用モータ

Google
ロボット用モーター
ロボット用モータ | シチズン千葉精密
ccj.citizen.co.jp
パナソニック、産ロボ用小型モーター開発...
nikkan.co.jp
ロボット用サーボモーター ...
prtimes.jp
ロボット用ハイトルクサーボ...
vstone.co.jp
円筒形モジュールMシリーズ
ロボット開発を容易にするオールインワンのモータ...
robotstart.info
Amazon | アーテック ロボット用DCモーター 15...
amazon.co.jp
ロボット関節用サーボモーター｜アダマ...
ad-na.com
ロボット用フレームレスモータ Kollmor...
npm-ht.co.jp
Amazon | Treedix 4個 TTモー...
amazon.co.jp
高トルクモーター(二足歩行ロボッ...
wowma.jp
回転
前後
上下
ひねり
曲げ
旋回
コアレスモータ 株式会社
小型ロボット用
ギヤ部
ギヤ内蔵モータ

モータの制御


入力
+
−
フィードバック
電圧
センサ
制御モジュール
DYNAMIXEL
XL-Series


大阪大学
OSAKA UNIVERSITY
OPEN 2021
Hosoda Lab.
Adaptive Robotics Group, Osaka University

初学者にやさしい教科書

角度の指令値

指令された角度

速度の指令値

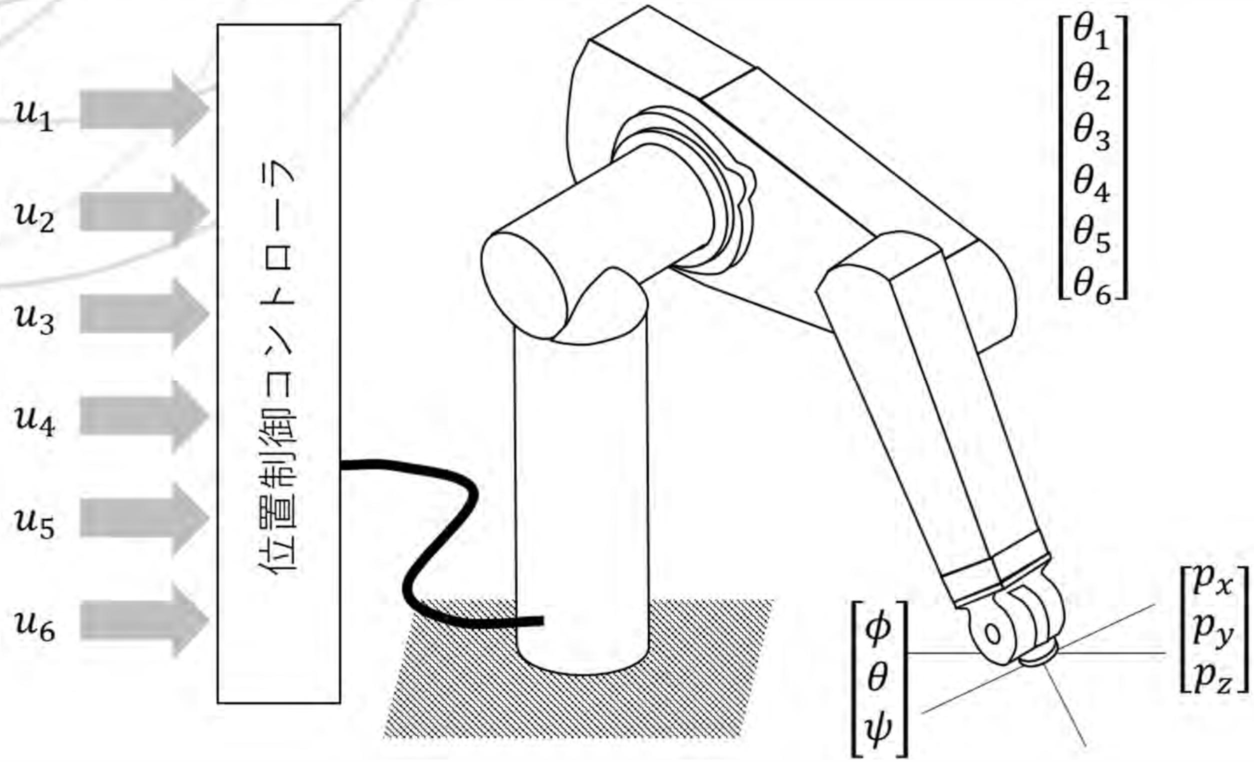
指令された速度

電流の指令値

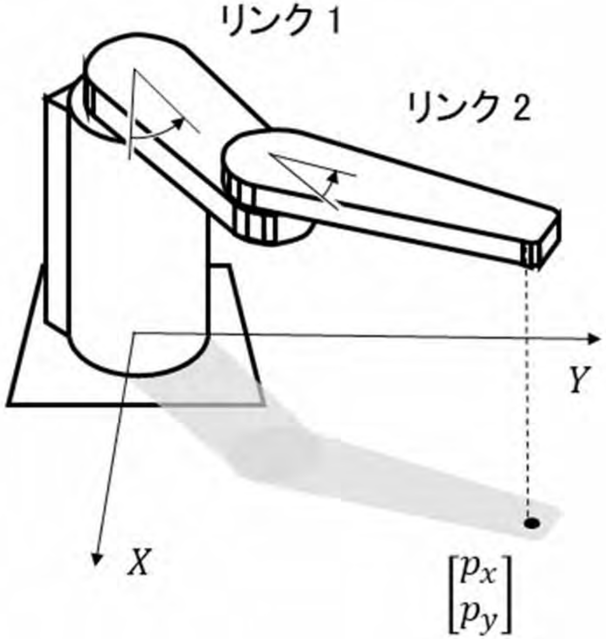
指令された回転力

レベルに応じて必要な知識が変わる

位置制御されたモータを使う場合

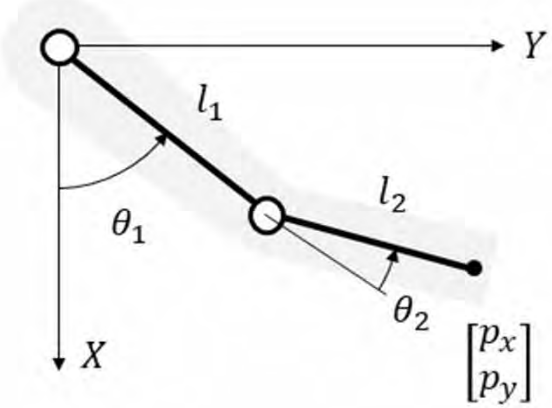


位置に関する逆運動学



順運動学

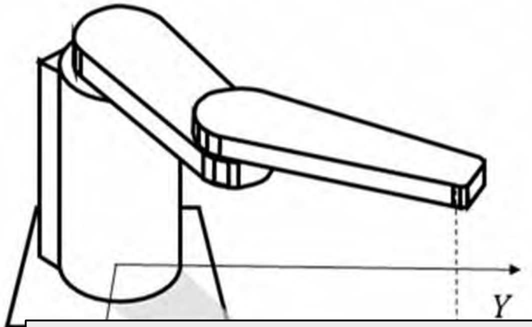
$$\boldsymbol{p}(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \end{bmatrix} = \begin{bmatrix} l_1C_1 + l_2C_{12} \\ l_1S_1 + l_2S_{12} \end{bmatrix}$$



$$\theta_1(t) = \mathrm{atan2}\left(-l_2S_2p_x(t) + (l_1 + l_2C_2)p_y(t), (l_1 + l_2C_2)p_x(t) + l_2S_2p_y(t)\right) \quad (1.9)$$

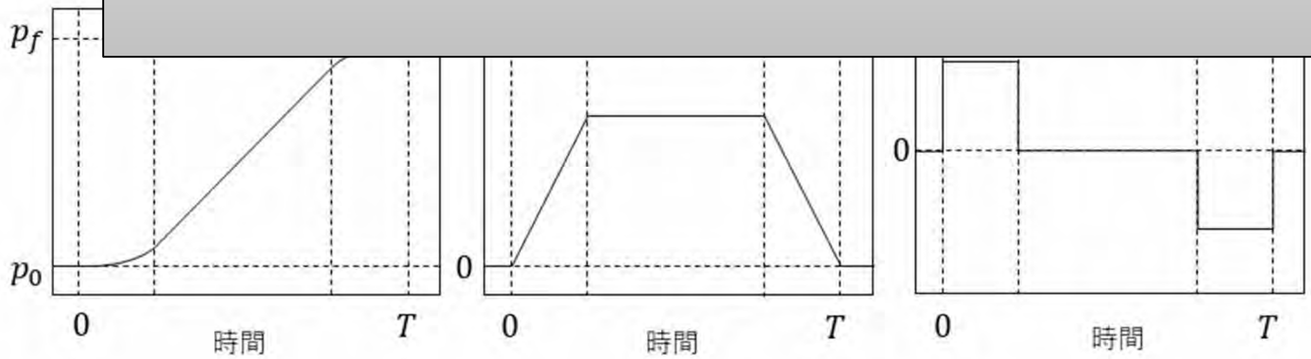
$$C_2 = \frac{p_x(t)^2 + p_y(t)^2 - {l_1}^2 - {l_2}^2}{2l_1l_2} \quad (1.3)$$

位置に関する軌道計画

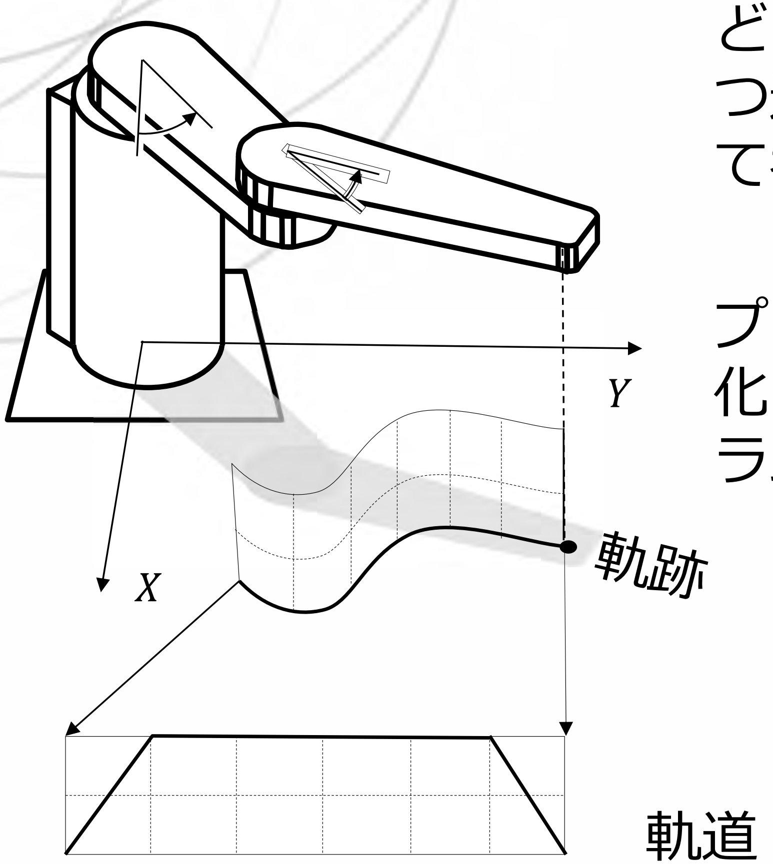


例え

ここまでの知識で,
２・３自由度の簡単なロボットは
思い通り動かせる



軌跡と軌道（こだわりポイント）



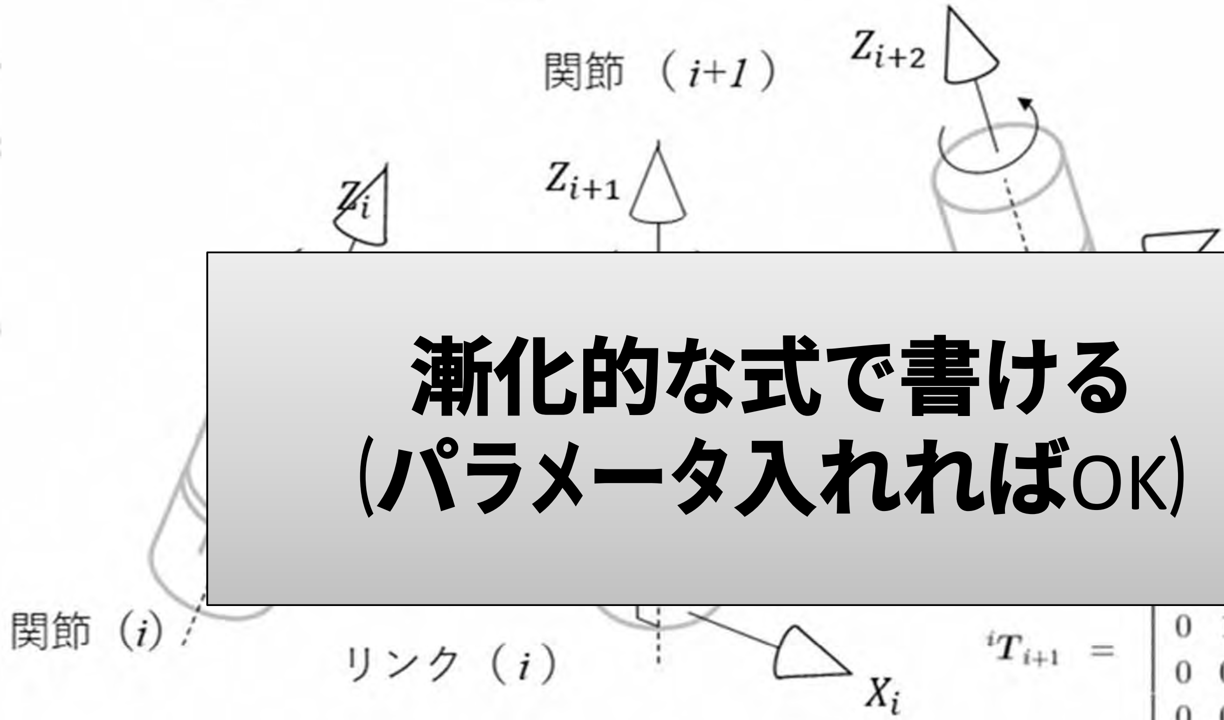
どこを通るかという「軌跡」といつ通るかという時間の要素を分けて考えることで，汎用性が高まる

プログラムするときにモジュール化しやすい（他人が使えるプログラムになりやすい）

多自由度・位置に関する順運動学

リンク座標系と
リンクパラメータ

リンク座標系とリンクパラメータ

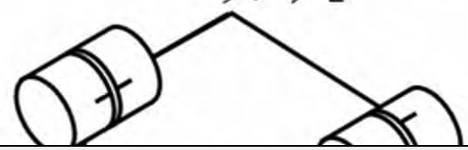


漸化的な式で書ける
(パラメータ入れればOK)

$$
{}^{0}\boldsymbol{T}_n = \left[\begin{array}{c|c} {}^{0}\boldsymbol{R}_n & {}^{0}\boldsymbol{p}_n \\ \hline 0\ 0\ 0 & 1 \end{array}\right] = {}^{0}\boldsymbol{T}_1\,{}^{1}\boldsymbol{T}_2\cdots{}^{n-1}\boldsymbol{T}_n
$$

$$
{}^{i}\boldsymbol{T}_{i+1} = \begin{bmatrix} \text{[illegible]} & & & \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_{i+1} & -\sin\theta_{i+1} & 0 & 0 \\ \sin\theta_{i+1} & \cos\theta_{i+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}
$$

多自由度・位置に関する逆運動学



「微分逆運動学」を知れば,
この式を「逆」に
解くことができるようになる

$${}^0T_n = \left[\begin{array}{c|c} {}^0R_n & {}^0p_n \\ \hline 0\ 0\ 0 & 1 \end{array}\right] = {}^0T_1\,{}^1T_2 \cdots {}^{n-1}T_n$$

これを展開して記号的に「逆」を計算するのは**絶望的**

「初学者にやさしい教科書」

大阪大学 OSAKA UNIVERSITY　OPEN 2021
Hosoda Lab. Adaptive Robotics Group, Osaka University

モータの制御レベルによって必要な知識が変わる
→「いつでもやめられるロボット工学」

平面2自由度のケースで直感的な理解をし，その後多自由度の説明に

平面2自由度など簡単な場合はそれでよし．多自由度を知るには微分運動学（次へ）

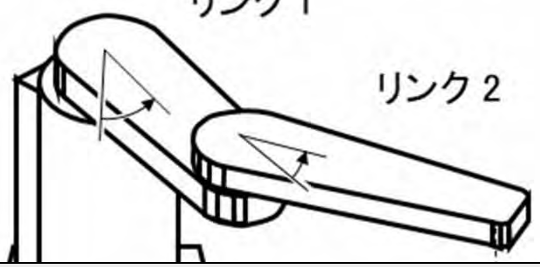
教科書の「しかけ」

初学者にやさしい教科書．でもしっかり理論は押さえる

コンピュータ科学，IoTの時代に即した知識を提供する

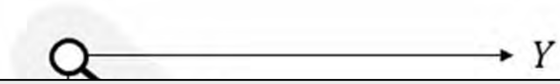
退屈な理論ではなく，数学・物理学が実践へとつながる

位置に関する逆運動学（再）



順運動学

$$p(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \end{bmatrix} = \begin{bmatrix} l_1C_1 + l_2C_{12} \\ l_1S_1 + l_2S_{12} \end{bmatrix}$$

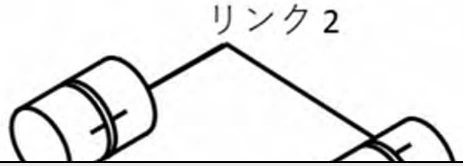


プログラムするためには
「解析的に」式を展開する必要がある

$\theta_1(t)$

コンピュータ技術が進んだ
「いま風」な方法を学ぼう！

多自由度・位置に関する逆運動学



プログラムするためには
「解析的に」式を展開する必要がある

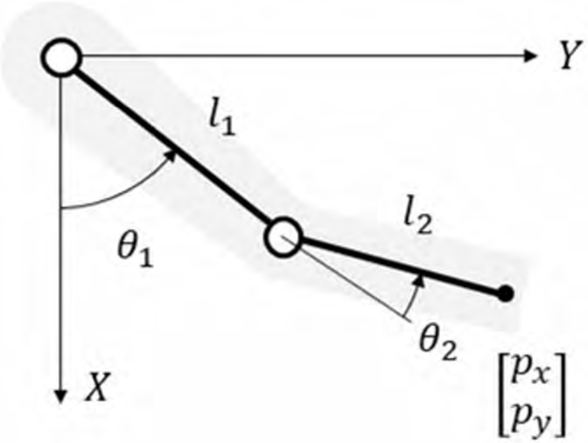
コンピュータ技術が進んだ
「いま風」な方法を学ぼう！

これ

ヤコビ行列の定義

$$\boldsymbol{r} = \boldsymbol{f}_r(\boldsymbol{q})$$

$$\begin{aligned}\dot{\boldsymbol{r}} &= \frac{\partial \boldsymbol{f}_r}{\partial q_1}\dot{q}_1 + \frac{\partial \boldsymbol{f}_r}{\partial q_2}\dot{q}_2 \cdots + \frac{\partial \boldsymbol{f}_r}{\partial q_n}\dot{q}_n \\ &= \left[\frac{\partial \boldsymbol{f}_r}{\partial q_1}\frac{\partial \boldsymbol{f}_r}{\partial q_2}\cdots\frac{\partial \boldsymbol{f}_r}{\partial q_n}\right]\dot{\boldsymbol{q}}\end{aligned}$$



例えばいつものこいつだと，

$$\begin{aligned} p_x &= l_1C_1 + l_2C_{12} \\ p_y &= l_1S_1 + l_2S_{12} \end{aligned}$$

この式の両辺を時間微分してベクトルの形にまとめると，

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \begin{bmatrix} -l_1S_1 - l_2S_{12} & -l_2S_{12} \\ l_1C_1 + l_2C_{12} & l_2C_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{6.6}$$

基礎ヤコビ行列＝ネ申！

基礎ヤコビ行列は,
解析的な微分を必要としない.
回転行列
を計算できれば導くことができる

$$
\begin{aligned}
\begin{bmatrix} {}^0\dot{\boldsymbol{p}}_E \\ {}^0\boldsymbol{\omega}_E \end{bmatrix} &= \begin{bmatrix} {}^0\boldsymbol{z}_1 \times {}^0\boldsymbol{p}_{E,1} & {}^0\boldsymbol{z}_2 \times {}^0\boldsymbol{p}_{E,2} & \cdots & {}^0\boldsymbol{z}_n \times {}^0\boldsymbol{p}_{E,n} \\ {}^0\boldsymbol{z}_1 & {}^0\boldsymbol{z}_2 & \cdots & {}^0\boldsymbol{z}_n \end{bmatrix} \dot{\boldsymbol{q}} \\
&= \boldsymbol{J}_v \dot{\boldsymbol{q}} \qquad (6.41)
\end{aligned}
$$

ヤコビ行列の解析的表現と漸化的表現

- 自由度が少ない時には, 手で計算ができる.
- 式が閉じた形で出てくるのでわかりやすい.
- 位置に関する逆運動学が面倒

解析的な表現

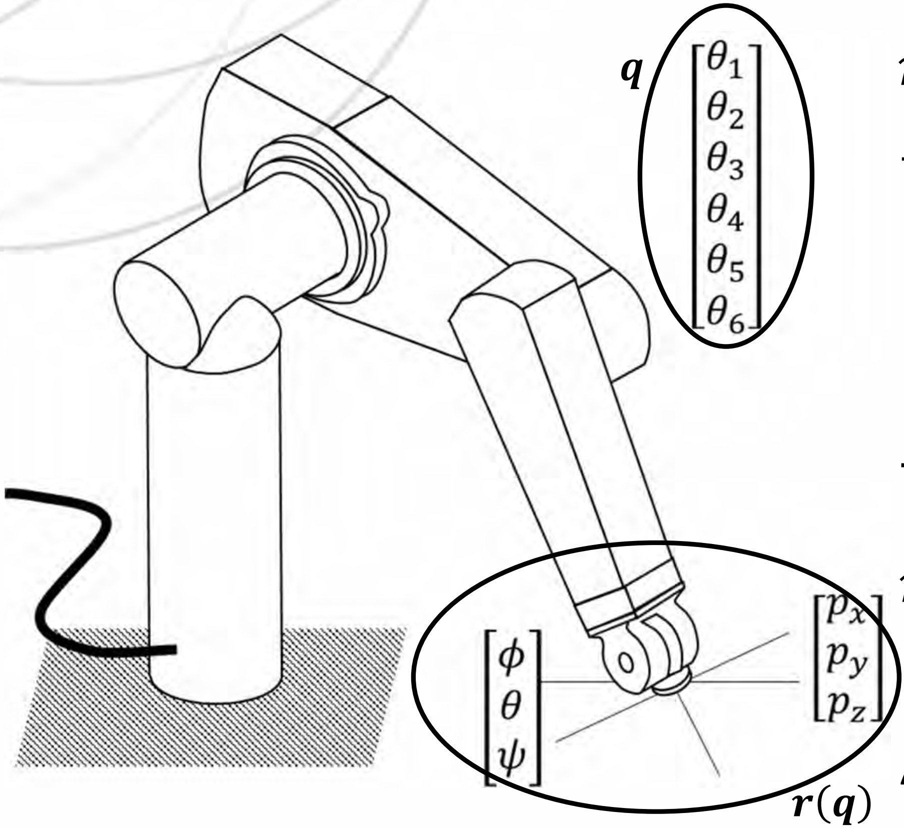
- 漸化式なので, 汎用モジュールにしやすい
- 直感的にわかりにくい
- 位置に関する逆運動学を数値的に解ける

$= \; J_v \dot{q}$

漸化的な表現

位置に関する逆運動学再考

与えられた $\boldsymbol{r}_d$ を実現する $\boldsymbol{r}(\boldsymbol{q})$ の $\boldsymbol{q}$ を求める問題



$\boldsymbol{r}(\boldsymbol{q}) - \boldsymbol{r}_d = \boldsymbol{0}$　を満たす $\boldsymbol{q}$ を求める問題

直接 $\boldsymbol{q}$ について解くのは難しい

真の解に近い $\overline{\boldsymbol{q}}$ があれば

$$\boldsymbol{r}(\overline{\boldsymbol{q}} + \Delta\boldsymbol{q}) - \boldsymbol{r}_d = \boldsymbol{0}$$

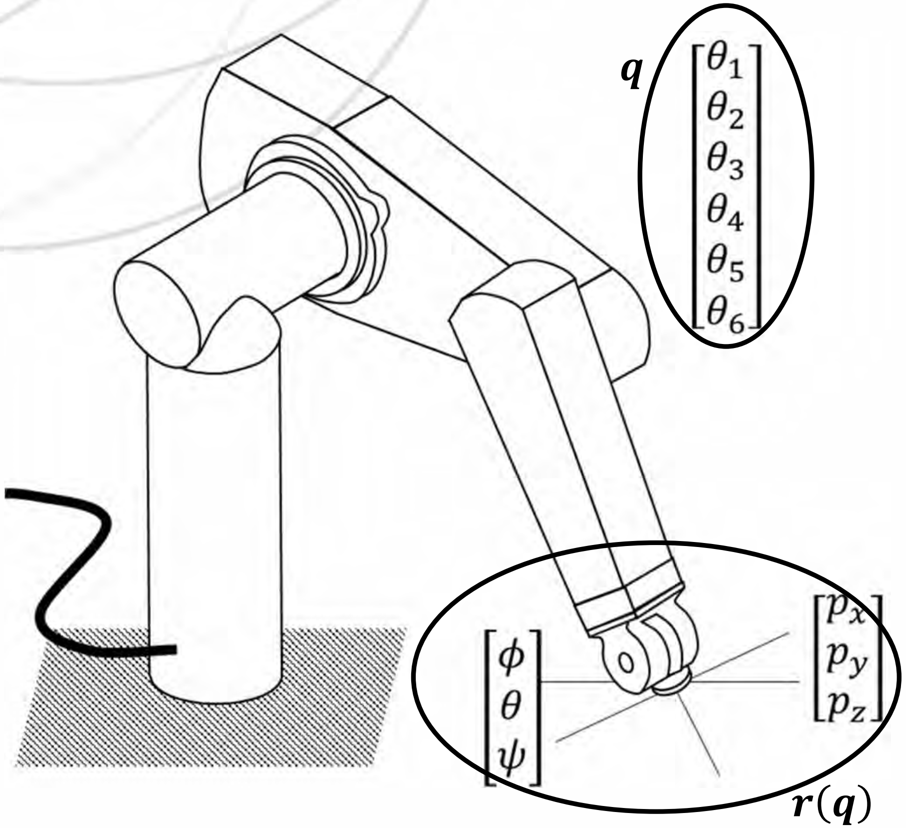
テーラー展開すると

$$\boldsymbol{r}(\overline{\boldsymbol{q}}) - \boldsymbol{r}_d + \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}^T}\Delta\boldsymbol{q} = \boldsymbol{0}$$

$$\Delta\boldsymbol{q} = \left[\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}^T}\right]^{-1}\{\boldsymbol{r}_d - \boldsymbol{r}(\overline{\boldsymbol{q}})\}$$

位置に関する逆運動学再考

与えられた $\boldsymbol{r}_d$ を実現する $\boldsymbol{r}(\boldsymbol{q})$ の $\boldsymbol{q}$ を求める問題



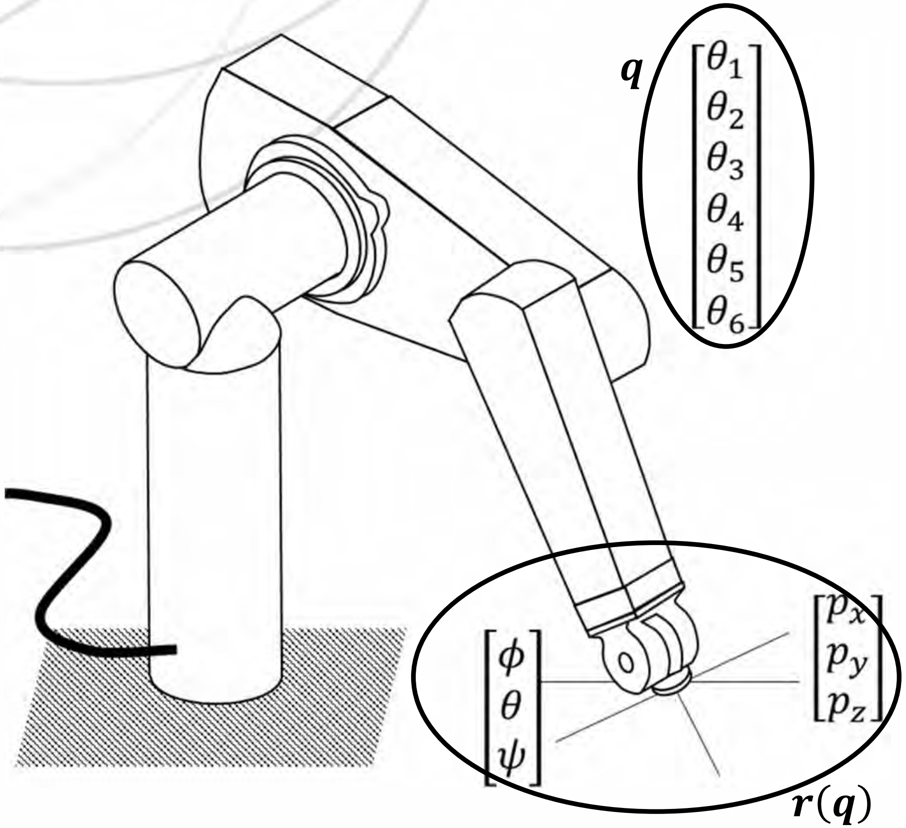
① $\boldsymbol{q}$の適当な初期値$\overline{\boldsymbol{q}}$を決める

② $\boldsymbol{r}_d - \boldsymbol{r}(\boldsymbol{q})$が十分小さければ終了．そうでなければ次の修正ステップへ

③ $\overline{\boldsymbol{q}} \leftarrow \overline{\boldsymbol{q}} + \left[\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}^T}\right]^{-1} \{\boldsymbol{r}_d - \boldsymbol{r}(\overline{\boldsymbol{q}})\}$
と修正，②に戻る

位置に関する逆運動学再考

与えられた $\boldsymbol{r}_d$ を実現する $\boldsymbol{r}(\boldsymbol{q})$ の $\boldsymbol{q}$ を求める問題



① $\boldsymbol{q}$の適当な 値$\bar{\boldsymbol{q}}$を る

② $\boldsymbol{r}_d - \boldsymbol{r}(\bar{\boldsymbol{q}})$が十分 れば終了．そうで ば次の修正ステップ

③ $\bar{\boldsymbol{q}} \leftarrow \bar{\boldsymbol{q}} + \left[\dfrac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}^T}\right]^{-1} \{\boldsymbol{r}_d - \boldsymbol{r}(\bar{\boldsymbol{q}})\}$

と修正，②に戻る

位置に関する順運動学

ヤコビ行列の逆

こいつらは
漸化的に解ける式

鬼門だった「位置に関する逆運動学」
がない！

大阪大学 OSAKA UNIVERSITY　OPEN 2021

Hosoda Lab. Adaptive Robotics Group, Osaka University

モータの制御

角度の指令値

指令された角度

速度の指令値

指令された速度

電流の指令値

指令された回転力

レベルに応じて
必要な知識が変わる

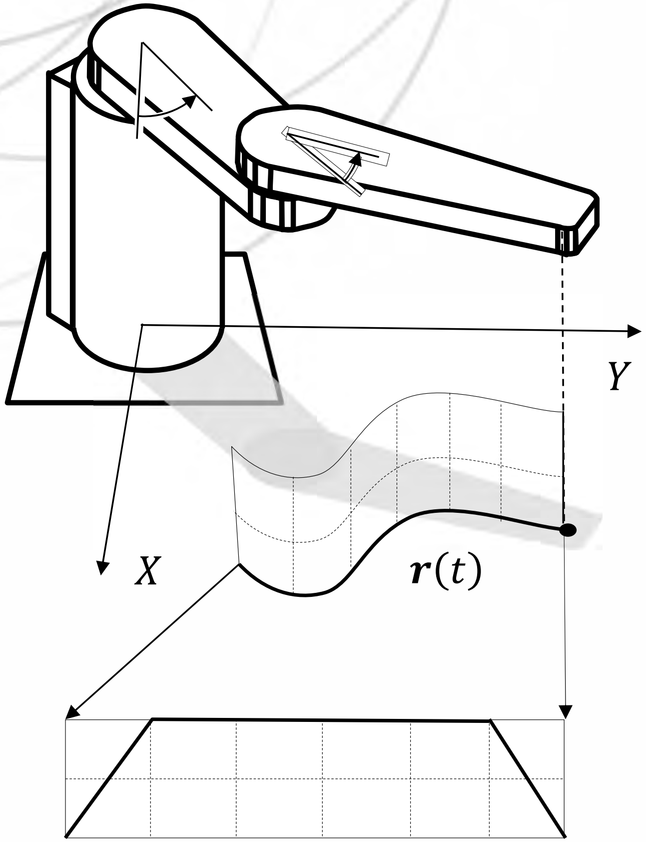
大阪大学 OSAKA UNIVERSITY

OPEN 2021

Hosoda Lab.
Adaptive Robotics Group, Osaka University

位置制御されたモータによる軌道制御

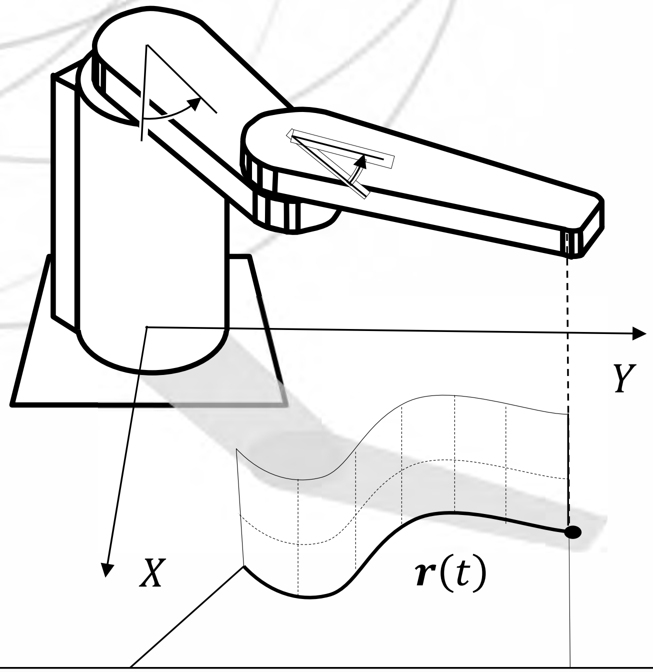
$\boldsymbol{r}(t) \rightarrow \boldsymbol{q}(t)$：位置に関する逆運動学問題



計算された $\boldsymbol{q}(t)$

各時刻でこの逆運動学問題を解かなければ！

速度制御されたモータによる軌道制御



手先の誤差が一次遅れ
で０になるとして

$$\dot{\boldsymbol{r}} - \dot{\boldsymbol{r}}_d + \boldsymbol{K}_p(\boldsymbol{r} - \boldsymbol{r}_d) = \boldsymbol{0}$$

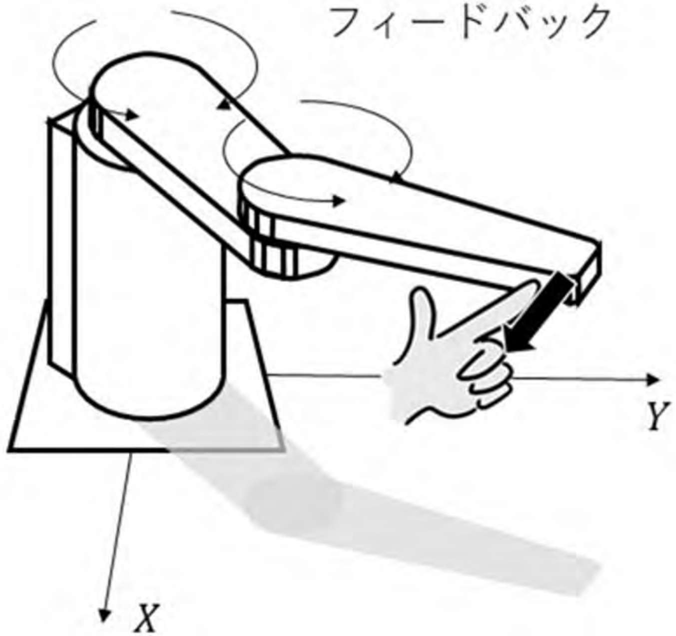
$$\boldsymbol{K}_p = \begin{bmatrix} k_{p1} & 0 & 0 & 0 \\ 0 & k_{p2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & k_{pn} \end{bmatrix}$$

$$\dot{\boldsymbol{r}} = \boldsymbol{J}_r \dot{\boldsymbol{q}}$$

$$\boldsymbol{J}_r \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_d - \boldsymbol{K}_p(\boldsymbol{r} - \boldsymbol{r}_d)$$

速度制御されたモータを使えば，
位置に関する逆運動学問題（とても面倒あるいは繰り返し計算必要）なし
順運動学と，ヤコビ行列の逆行列だけで計算が済む

コンプライアンス制御（こだわりポイント）



各軸フィードバック

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = -\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix}$$

だと，手先の特性はこうなる

$$\begin{bmatrix} f_x \\ f_y \end{bmatrix} = -\boldsymbol{J}^{-T} \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \boldsymbol{J}^{-1} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

あらかじめ，

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = -\boldsymbol{J}^{T} \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \boldsymbol{J} \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix}$$

としておけば，

$$\begin{bmatrix} f_x \\ f_y \end{bmatrix} = -\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

コンピュータ科学の時代に即した内容

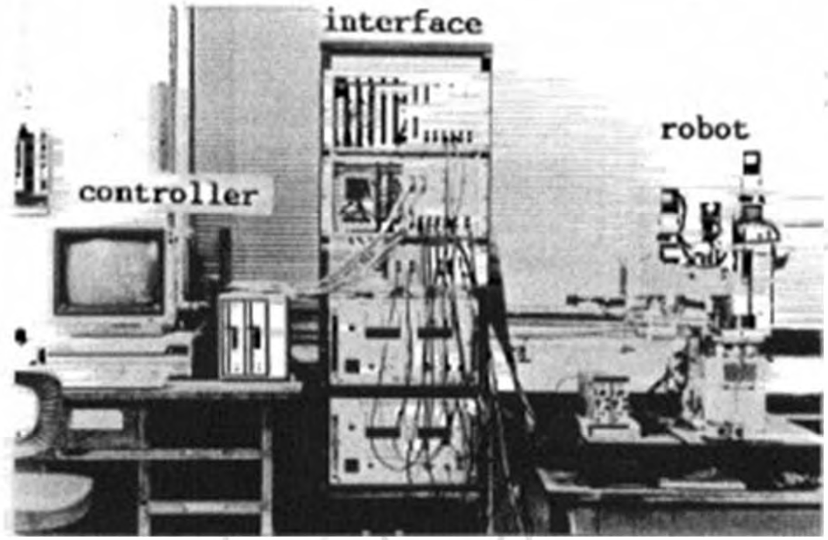
解析的に解けるならわかりやすいんだけど,「位置に関する逆運動学問題」は難しい

基礎ヤコビ行列を使うと,漸化的表現だけで「位置に関する逆運動学問題」が解ける

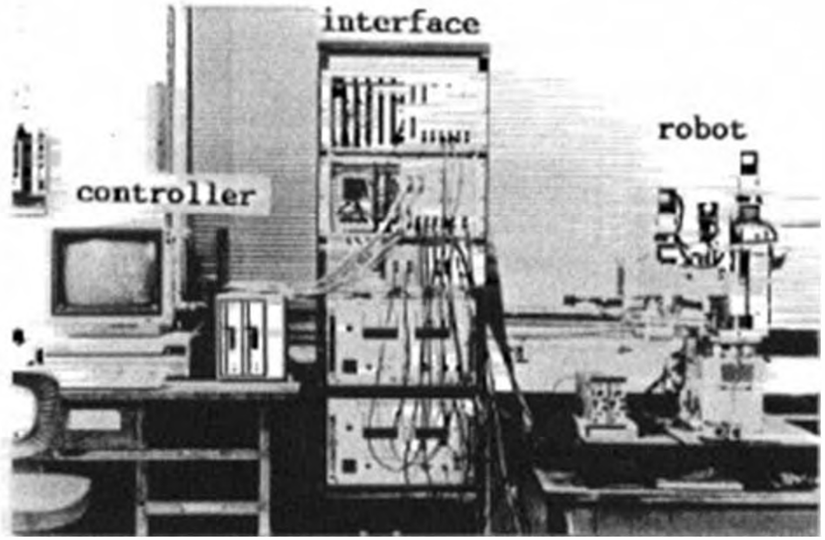
手先の軌道制御も,位置に関する逆運動学問題を解かずに,ヤコビ行列と順運動学のみですむ

ロボット制御は面白くない（当時）

- ヤコビ行列ってなに？なんの役に立つの？
- シミュレーションするには運動方程式は必要
- ラグランジュの方法で運動方程式出せたらそれでいい
- 各軸にPID制御使えば十分精度出るし，動的制御なんて現場ではいらんのとちゃう？（実際1990年代はそうだった）



電流制御こそ王道！（当時）



- 動的制御するには，電流制御できないと意味ない！
- 動的制御は計算量が問題（計算の並列化なども研究になった）．

すべて「コンピュータ技術」が
解決してしまった

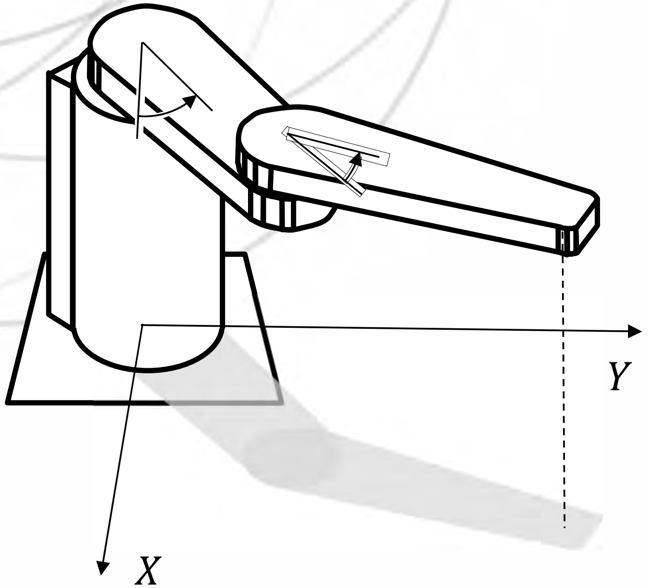
教科書の「しかけ」

初学者にやさしい教科書．でもしっかり理論は押さえる

コンピュータ科学，IoTの時代に即した知識を提供する

退屈な理論ではなく，数学・物理学が実践へとつながる

ロボットの運動方程式



Lagrange の運動方程式

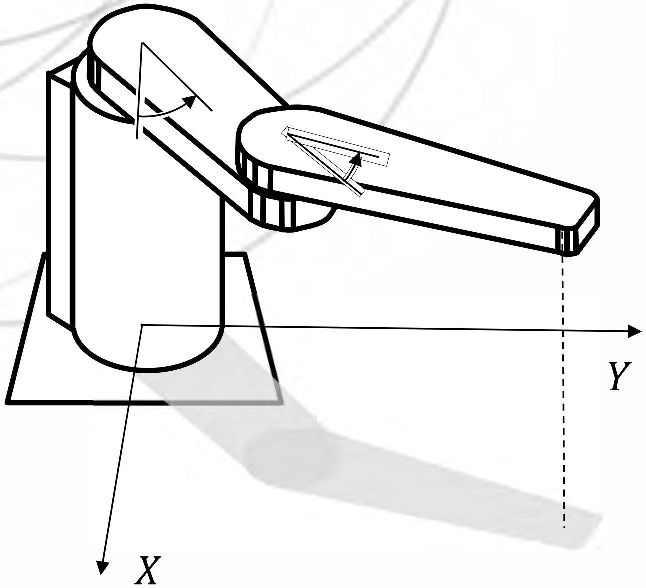
偏微分を伴う，解析的計算に基づく導出

Newton-Euler の運動方程式

漸化的計算に基づく導出

実はこんなことはどうでもいいんです！

ロボットの運動方程式



もっとも重要なのは

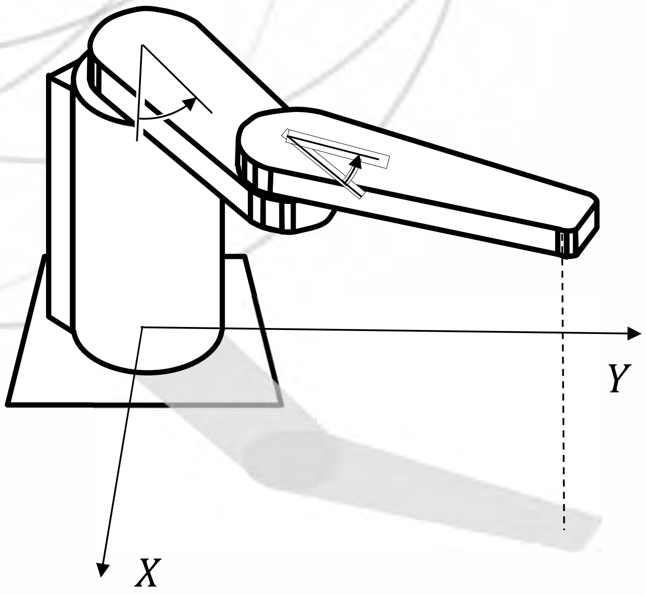
どんなロボットでも，どんな方法を使っても，運動方程式は

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

という形になること

ロボットの運動方程式

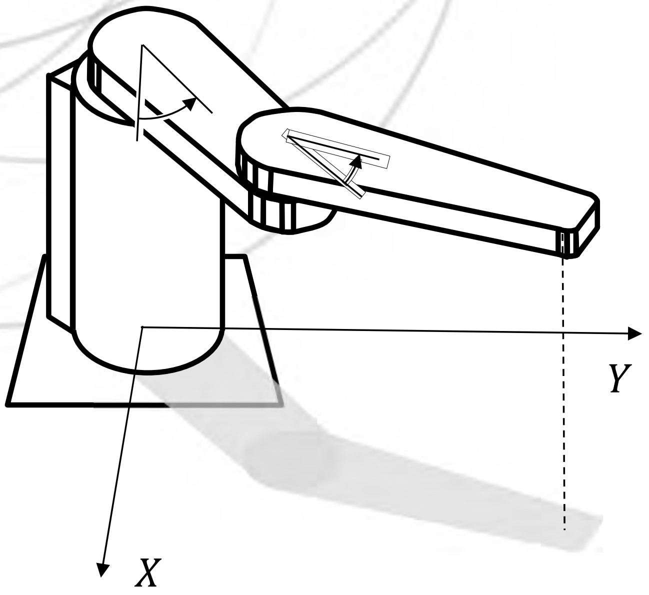
大阪大学
OSAKA UNIVERSITY
OPEN 2021
Hosoda Lab.
Adaptive Robotics Group, Osaka University



$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

にいろいろな制御をツッコむと，
何が起こるかを予測できる

各軸フィードバックの安定性



$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$$

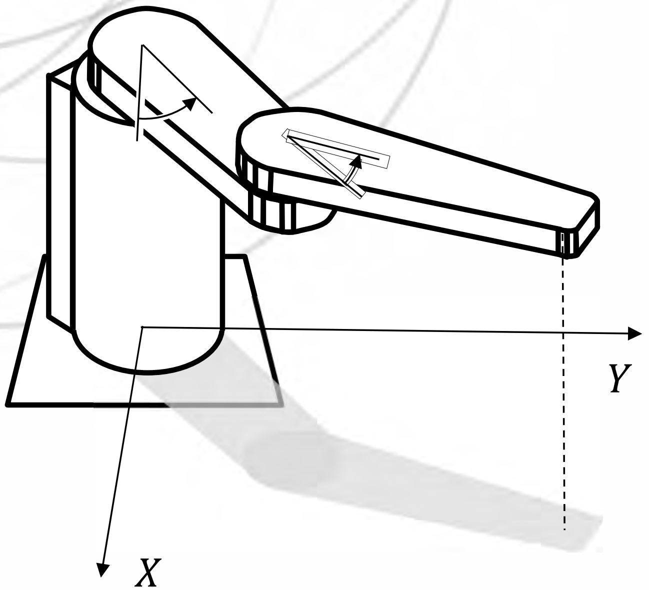
にいろいろな制御をツッコむと，
何が起こるかを予測できる

各軸フィードバック制御

$$\boldsymbol{\tau} = \boldsymbol{K}_p(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{K}_v\dot{\boldsymbol{q}}$$

は，モータの減速比とフィード
バックゲインが十分に大きければ
安定

各軸フィードバックの安定性



$$M(q)\ddot{q} + h(q,\dot{q}) + g(q) = \tau$$

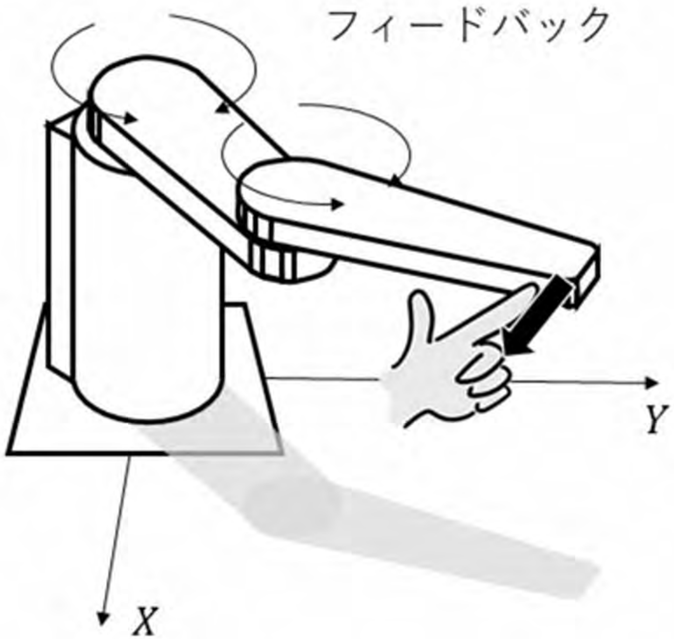
にいろいろな制御をツッコむと，
何が起こるかを予測できる

各軸フィードバック制御
+重力補償

$$\tau = K_p(q_d - q) + K_v\dot{q} + g(q)$$

は，無条件に安定

コンプライアンス制御も



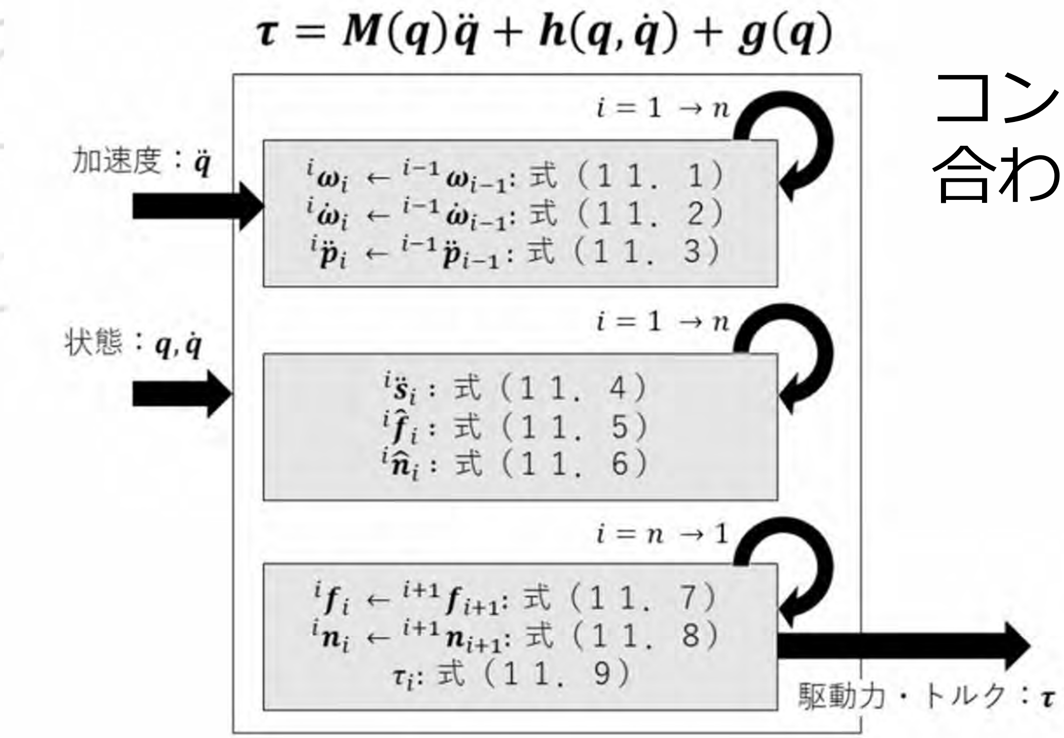
$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = -J^T \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} J \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix}$$

$$\boldsymbol{\tau} = -\boldsymbol{J}^T \boldsymbol{K} \boldsymbol{J} \Delta \boldsymbol{q} + \boldsymbol{g}(\boldsymbol{q})$$

を使うことで，重力下の安定性を保証することができる

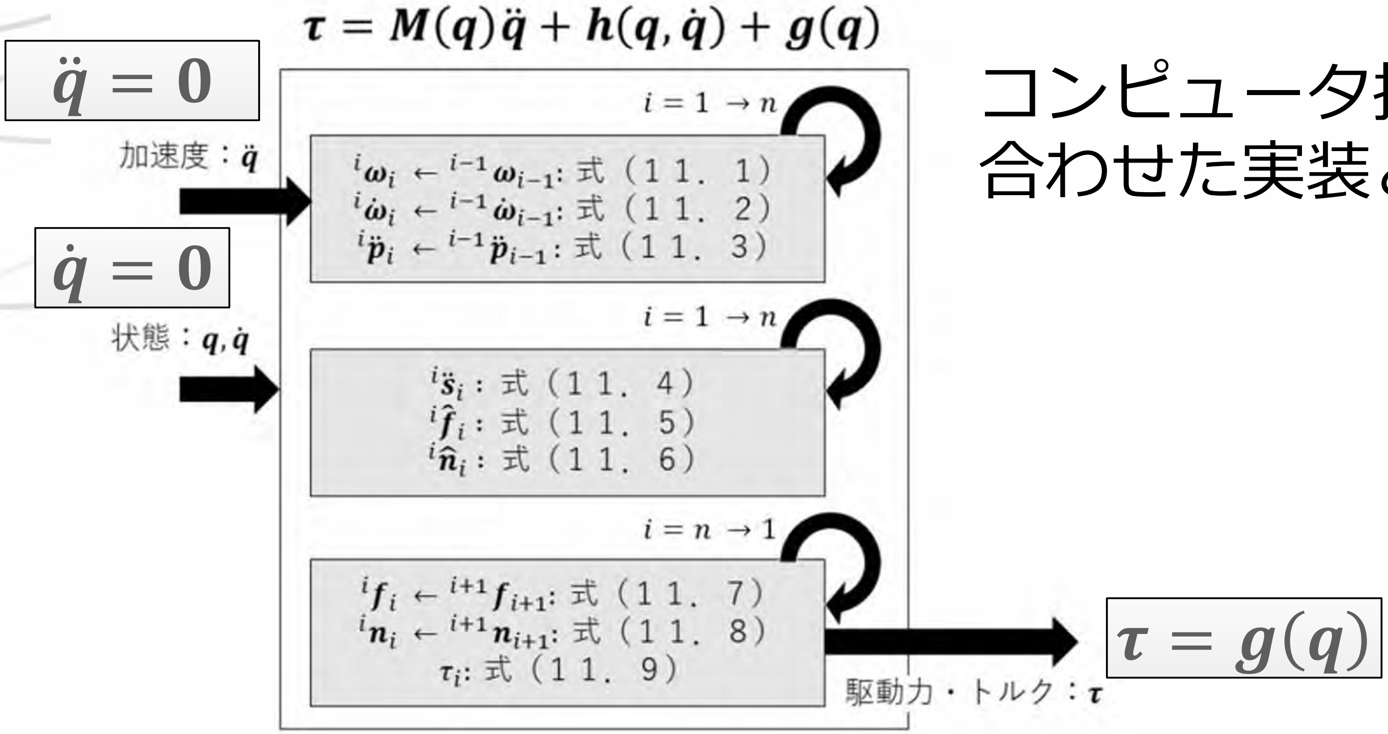
モジュールとしての運動方程式

大阪大学 OSAKA UNIVERSITY
OPEN 2021
Hosoda Lab. Adaptive Robotics Group, Osaka University



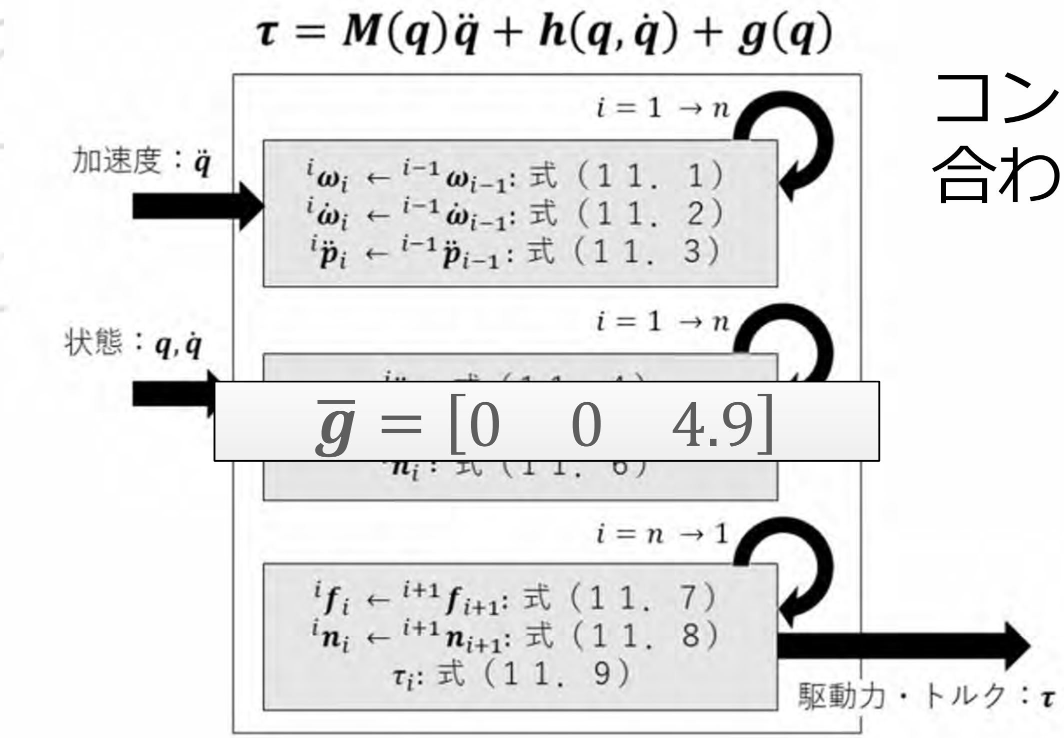
コンピュータ技術に
合わせた実装と運用

重力補償だけを取り出す



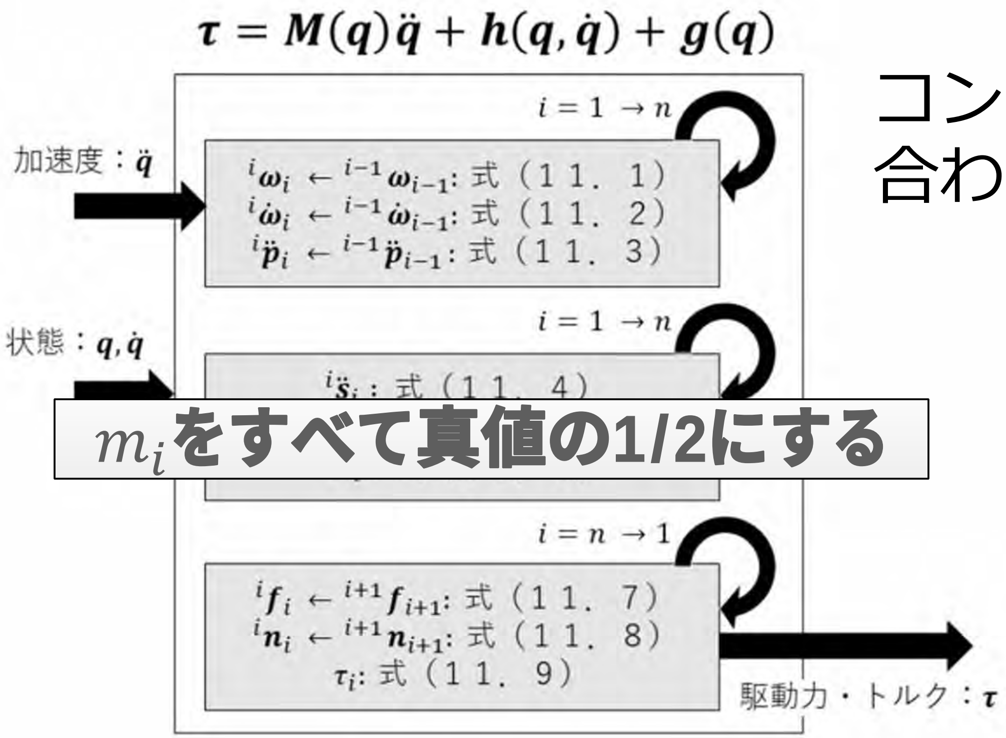
コンピュータ技術に
合わせた実装と運用

重力を半分だけ補償する



コンピュータ技術に
合わせた実装と運用

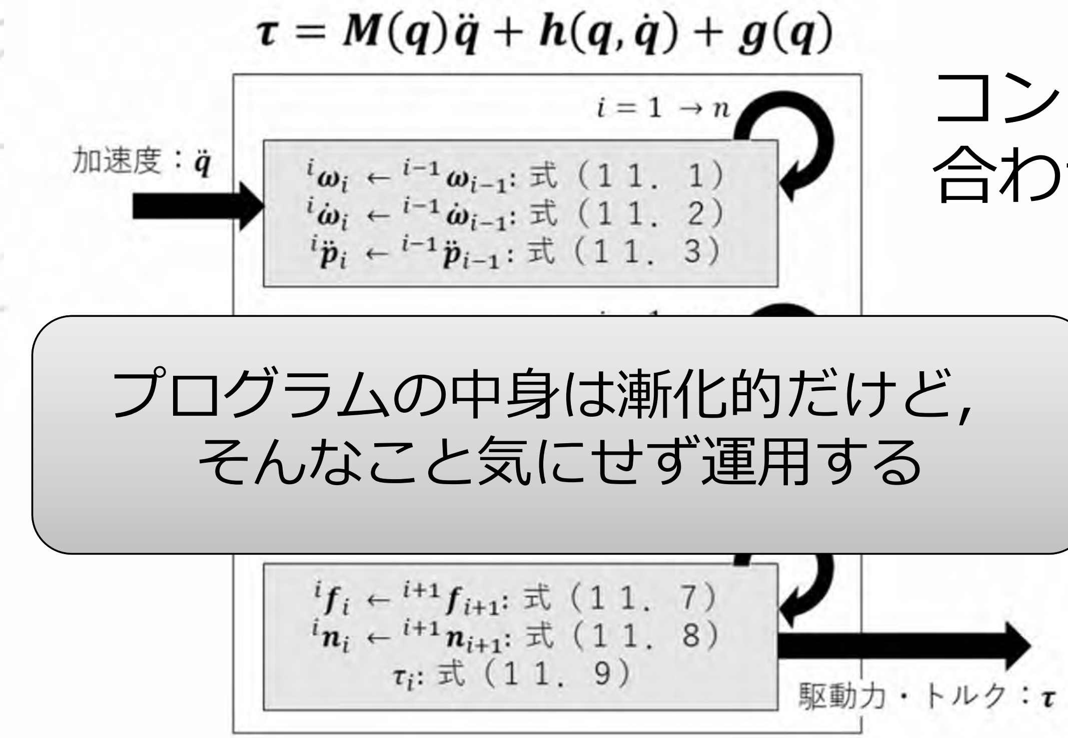
リンクの重量を半分だけ補償する



m_iをすべて真値の1/2にする

コンピュータ技術に
合わせた実装と運用

モジュールとしての運動方程式



プログラムの中身は漸化的だけど，
そんなこと気にせず運用する

コンピュータ技術に
合わせた実装と運用

数学・物理学と実践との橋渡し

運動方程式自体の実装はどうでもよい（どうでもよくない）

運動方程式の性質を見据えた「動的制御」を考える

退屈な理論ではなく，数学・物理学が実践へとつながる

ロボット工学を教えよう！習おう！

初学者にやさしい教科書．でもしっかり理論は押さえる

コンピュータ科学，IoTの時代に即した知識を提供する

退屈な理論ではなく，数学･物理学が実践へとつながる