オブジェクト指向による位相面解析*

大 野 邦 夫^{*1}

Phase Plane Analysis based on Object-Oriented Approach

Kunio OHNO

Through phase plane analysis, a nonlinear system is defined on the phase plane, and the solution is obtained as a trajectory. Based on the idea that the relationship between the nonlinear system and its solution could be compared with the relationship between class and instance of object-oriented systems, a simulation program for the phase plane analysis was written in the programming language TAO. Simple oscillation, damped oscillation, and self-sustained oscillation of Van der Pol equation, Rayleigh equation, and the stick-slip oscillation were implemented in that program. Compared with the existing functional programming approach, the effectiveness of object-oriented phase plane analysis for nonlinear vibration was discussed and proven through the program.

Key Words: Phase Plane, Object-Oriented, Nonlinear Vibration, Stick Slip, Simulation

1. まえがき

本論文は、2次元のオートノマスな非線形系の解法で用いられる位相面解析法⁽¹⁾⁽²⁾にオブジェクト指向技術を適用し、その有効性を示すものである。位相面とは、対象となる変数とその時間に関する導関数を座標軸にする平面で、振動系の場合であれば、変位と速度が座標軸になる。ある特定の振動系を対象に考えると、振動系の微分方程式すなわち状態方程式は、位相面上の各点で解曲線（トラジェクトリ）の方向を定めるものであり、初期値を与えることにより、位相平面上の具体的なトラジェクトリとして解を得ることができる。

ところで、以上における、振動系とトラジェクトリとの関係は、オブジェクト指向におけるクラスとインスタンスの関性に類似している。すなわち、振動系とは質量や変位、速度等を変数とする抽象的な対象であり、その系に特有な状態方程式を持つものである。そしてその解は、初期値から定まる具体的な状態変化の一例である。そこで、振動系をクラスとして定義し、状態方程式をそのクラスに属するメソッドとして定義し、個別のトラジェクトリをインスタンスとして扱う手法

が考えられる。

ここでは、以上の考え方にに基づき、単振動、減衰振動、自励振動、摩擦振動について位相面解析し、オブジェクト指向の適用が効果的であることを述べる。

なお、振動的な現象の解析に、オブジェクト指向を適用した従来の研究もあるが、それらは波形の解析システムの構築⁽³⁾や定性推論のモデルのための一例⁽⁴⁾などであり、本研究のように、非線形振動系そのものを対象にオブジェクト指向を用いて解いているものは、見当たらない。

また、各種分野で、システムをモデル化して解析する場合に、最終的に非線形振動系の解明に到達する例が少なからず存在する⁽⁵⁾⁽⁶⁾⁽⁷⁾。そのような場合に本手法を有効に用いることができると思われる。

2. 計算機環境

2.1 プログラミング言語 NTTが開発したLISPマシンELIS上のプログラミング言語TAOを用いた⁽⁸⁾⁽⁹⁾。TAOはLISPの方言であるが、オブジェクト指向、ロジックプログラミング機能も融合させたマルチパラダイム言語である。オブジェクト指向においては、クラス定義、メソッド定義、インスタンス生成、メッセージ送出という手順でプログラムを実行する。TAOの場合

* 原稿受付 平成5年11月16日。

*1 正員, NTTヒューマンインタフェース研究所 (〒238 横須賀市武1-2356)。

について具体的に示すと以下ようになる。

2.2 クラス定義 defclassというマクロを用い以下のように定義される。

```
(defclass class-name
  class-variable-list
  instance-variable-list
  &opt super-class-list
  &rest options)
```

2.3 メソッド定義 クラスには固有の手続きを付属させることが可能である。その手続きをメソッドと呼び、以下のように関数defmethodで定義される。

```
(defmethod (class-name message-name)
  (argument-list) form)
```

2.4 インスタンス生成 make-instanceという関数を用いて次のように生成される。

```
(setq instance-name (make-instance
  'class-name &rest name-value-pairs))
```

2.5 メッセージ送出 以下に示す構文を用いる。
[instance-name message-name &rest arguments]
かぎ括弧はメッセージパッシングを表現するTAO独特の表記法である。TAOのメッセージパッシングは高速で、関数呼び出しと同等の性能を実現している⁽⁹⁾。

3. 非線形振動への適用

3.1 基本クラス

3.1.1 基本的考え方 オブジェクト指向プログラミングでは、まず基本クラスを定義し、それを継承する形で、下位のクラスを設定する。ここでは振動現象を対象するので、基本クラスとしては、最も単純な振動系であるばね質量系を取り上げる。運動方程式は、下記のように示される。

$$m(d^2x/dt^2) + kx = 0 \quad (1)$$

$k/m = \omega^2$ とし、 $s = \omega t$ とすると、(1)式は無次元化される。

$$d^2x/ds^2 + x = 0 \quad (2)$$

$dx/ds = y$ とすると、(2)式は、 $dy/ds = -x$ となる。ここで、 s を再び t と置き直す。

$$dx/dt = y \quad (3-1)$$

$$dy/dt = -x \quad (3-2)$$

(3)式は、ばね質量系の状態方程式である。この系の状態は、変位 x と速度 y を変数とする xy 平面上の点で表される。この点を状態点または状態ベクトルと言う。

状態ベクトルを v とすると、(3)式は、 $dv/dt = f(v)$ という一階のベクトル微分方程式の一例である。この式の意味は、「状態点 v における時間微分(状態速度)はその点の関数 $f(v)$ により定まる」ということである。このことから、

(1) ばね質量系の状態方程式(3)式の解は、初期値を与えるとその状態点における状態速度が定まり、微小時間後の新たな状態点を算出することが可能となる。

(2) 算出された新たな状態点をもとに、次の微小時間後の状態点の算出が可能となるので、これを繰り返せば位相面上の解曲線(トラジェクトリ)が得られる。

なお、この解法は、数学的には、オイラー法による微分方程式の数値解法に相当する。オブジェクト指向プログラミングは、位相面上の状態方程式およびその解曲線の記述に適している。すなわち、対象とする系をクラスとして定義し、状態変数をインスタンス変数として扱い、さらに状態方程式とそのオイラー法による解法をメソッドで記述する。

解を得るためには、インスタンスに初期値を与え、微小時間後の状態点を逐次求めてゆくが、これらはメッセージを通じて実行される。この場合には、状態変数としてのインスタンス変数が、メッセージにより逐次書き換えられ、状態点の軌跡が得られる。これが求める状態方程式の解である。以上の考え方に基づき、ばね質量系の微分方程式を解析する。

3.1.2 クラス定義 前項で述べたばね質量系(単振動)モデルのクラスを定義する。

```
(defclass simple-oscillation ()
  ((x 1.0) ;状態変数 (変位)
   (y 0.0) ;状態変数 (速度)
   (time 0.0) ;時間変数
   (dx dy (dt 0.01)) ;増分
   () ;スーパークラスなし。
  :gettable :settable)
```

3.1.3 メソッド定義 まず、(3)式をオイラー法により計算するメソッドを定義する。

```
(defmethod (simple-oscillation diff-eqn) ()
  (setq dx [y * dt]
    (setq dy (* -1 x dt)))
```

次に、変位の時間的な変化をウインドウ上に描くメソッドtransientを定義する。

```
(defmethod (simple-oscillation transient) ()
  (loop (:until ((> time max-time) nil)
    [self diff-eqn])
```

```
[self time-chart]
[self diffnc] ))
```

さらに、位相面上のトラジェクトリを描くメソッド trajectory を定義する。

```
(defmethod (simple-oscillation trajectory) ()
  (loop (until (> time max-time) nil)
    [self diff-eqn]
    [self phase-chart]
    [self diffnc] ))
```

なお、time-chart、phase-chart は、結果を時間現象および位相平面のウィンドウにプロットするためのメソッドである。diffnc は、次の段階の計算のために、x、y、time を増加させるためのメソッドである。

3.1.4 インスタンス生成 anExample という名称のインスタンスを生成する。

```
(setq anExample (make-instance 'simple-oscillation))
```

初期条件は、 $x=1$, $y=0$ すなわち、変位 x が 1 で、初速度が 0 である。

3.1.5 メッセージ送出 このプログラムを実行させるには、インスタンスにメッセージを送る必要がある。変位の時間的变化を表示させるには、[anExample transient] とキーボードから入力すればよい。結果を図 1 に示す。

図 1 単振動の時間的变化

単振動なのでサイン波形になる。次に位相面上のトラジェクトリを求めるためには、[anExample trajectory] とキーボードから入力する。この場合には図 2 のように楕円になる。なお、完全な楕円にならず螺旋になっているのは、オイラー法による数値解析の誤差に起因している。

図 2 単振動の位相面図

3.2 減衰振動モデル 減衰振動モデルは、単振動モデルに正の減衰項が加わったものである。そこで単振動のクラスを継承することにより減衰振動のクラスを定義する。

```
(defclass damped-oscillation ()
  ((damping-factor 0.1)
   (simple-oscillation)
   :gettable :settable)
  (defmethod (damped-oscillation diff-eqn) ()
    (setq dx [y * dt])
    (setq dy
      (* -1 [(^ 2.0 damping-factor y) + x] dt)))
```

インスタンス生成およびメッセージ送出は、単振動の場合と同様である。各種 damping-factor について実行した時間現象の波形を図 3 に示す。オブジェクト指向の継承機能を用いているので、クラスを定義して簡単なメソッドを定義するだけで、その他のことについては、スーパークラスである単振動の時と全く同様な処理が可能である。

図 3 各種減衰係数についての減衰振動波形

3.3 Van der Polの式 自励振動のモデルとして有名な、Van der Polの方程式⁽¹⁰⁾⁽¹¹⁾をモデルとして記述する。

$$d^2x/dt^2 + \mu(x^2 - 1)(dx/dt) + x = 0 \quad (4)$$

連立の一階微分方程式にすると

$$dx/dt = y \quad (5-1)$$

$$dy/dt = -x - \mu(x^2 - 1)y \quad (5-2)$$

この式をVan der Polのクラスとして定義する。

```
(defclass Van-der-Pol ()
  ((mu 0.1))
  (simple-oscillation)
  :gettable :settable)
(defmethod (Van-der-Pol diff-eqn) ()
  (setq dx [y * dt]
        dy (* -1 [(mu (- (* x x) 1) y) + x] dt)))
```

自励振動が成長してゆく様子を見るために、(setq anExample (make-instance 'Van-der-Pol x 0.01))としてインスタンス生成し、初期値を位相面の原点近くに設定する。メッセージ送出は、単振動の場合と同様にすればよい。 $\mu=0.1$ の場合についての波形を図4に示す。

図4 Van der Pol 方程式の自励振動波形 ($\mu=0.1$)

図5 Van der Pol 方程式の位相面図 ($\mu=0.1$)

自励振動が成長し飽和してゆく様子がよく分かる。トラジェクトリを図5に示すが、原点近くからスパイラルを描いて振動が成長し、リミットサイクルを描いてゆく様子が理解できる。オブジェクト指向の継承機能を用いているので、減衰振動の場合と同様に、クラスを定義して簡単なメソッドを定義するだけで、その他のことについては、スーパークラスである単振動の時と全く同様な処理が可能である。さらに、

```
(setq anExample
  (make-instance 'Van-der-Pol x 0.01 mu 1.0))
```

として、[anExample trajectory]とすると、図6に示すように $\mu=1$ の場合のトラジェクトリを得ることができる。このように、インスタンス変数の値を変更することにより、パラメタの影響を容易に把握することができる。

図6 Van der Pol 方程式の位相面図 ($\mu=1.0$)

3.4 一般的な自励振動モデル 自励振動のモデルは、前項のVan der Polの方程式が有名だが、ここでは、拡張性を持たせた、より一般的な非線形抵抗を有する運動方程式をモデルとして記述する。

$$d^2x/dt^2 + f(dx/dt) + x = 0 \quad (6)$$

$f(dx/dt)$ が非線形項であり、この値が負だと発振し、正だと減衰する。この形の式は、Lienardの図式解法⁽¹²⁾⁽¹³⁾が適用可能である。連立の一階微分方程式にすると

$$dx/dt = y \quad (7-1)$$

$$dy/dt = -f(y) - x \quad (7-2)$$

この式をLienardのクラスとして定義する。

```
(defclass Lienard ()
  ()
  (simple-oscillation)
  :gettable :settable)
```

メソッドdiff-eqnは、このクラスのものとして定義す

る。同一名のメソッドが、スーパークラスにあるときは、自分のクラスのメソッドが優先して適用される。

```
(defmethod (Lienard diff-eqn) ()
  (setq dx [y * dt])
  (setq dy [self f - x] * dt))
```

(7・2)式の $f(y)$ は、 f というメソッドで定義される。ここでは、 $f(y)$ をRayleighの方程式⁽¹⁰⁾⁽¹³⁾としてシミュレートする。そのために、Rayleighというサブクラスを定義する。

```
(defclass Rayleigh ()
  ((mu 0.1)
   (Lienard)
   :gettable :settable)
  (defmethod (Rayleigh f) ()
    [mu * [y - [(y y y) / 3.0]]])
```

シミュレーション結果を図7に示す。ここでは μ を1.0とした場合を示している。

図7 Rayleigh方程式の位相面図 ($\mu=1.0$)

図式解法で用いるLienard曲線⁽¹²⁾⁽¹⁴⁾を描くメソッドも用意しており、位相平面上に描くことができる。

```
(defmethod (Lienard draw-f) (&aux x1)
  (loop (init (setq y y-min))
        (setq x [self f] x1 x)
        [self phase-chart]
        (setq y [y + dy]
              dx [[self f] - x1])
        (:until [y > y-max]))
```

4. 摩擦振動への適用

4.1 摩擦振動 摩擦振動とは、静止摩擦と動摩擦との差に起因して生ずる振動で、スティックスリップ振動とも呼ばれている。ドアの開閉の際のきしむ音、電車や自動車、自転車のブレーキの音、バイオリンや

チェロ等の弦楽器の音などは、摩擦振動による音である。一般に、摩擦によって生じる力は、相対すべり速度の関数として与えられる。一般的な速度摩擦特性を示すと、図8のようになり、基本的には3種類の要因に分けられる⁽¹⁵⁾。

- (1)乾性摩擦：速度に対して一定の摩擦力になる。
- (2)粘性摩擦：速度に比例した摩擦力となる。
- (3)境界摩擦：速度が増すと静止摩擦が急に低下し、後はゆるく減少する。

図8 速度摩擦特性

図9 摩擦振動のモデル

ここで、図9に示すような1自由度のばね質量系(質量 m 、ばね定数 k)において、質量 m の物体が一定速度 V で走行する粗いベルト上に置かれている場合を考える。物体の静止している位置を原点とし、移動台の移動方向に x 軸をとる。時刻 t における物体の位置を x とすると、物体と移動台との相対速度 U は、 $V - dx/dt$ となり、移動台が物体に及ぼす摩擦力は、 $F(V - dx/dt)$ となる。一般に、この摩擦力は相対速度 U に関する奇関数で、 $U > 0$ において $F(V - dx/dt)$ ならば、 $U < 0$ において $F(V - dx/dt)$ 、すなわち $-F(dx/dt - V)$ である。図9に示す系の場合、運動方程式は

$$m(d^2x/dt^2) + kx = F(V - dx/dt) \quad (8)$$

これは(1)式の右辺に摩擦力が加わった式である。無次元化すると。

$$d^2x/ds^2 + x = f(v - dx/ds) \quad (9)$$

なお、

$$f(v - dx/ds) = F(V - dx/dt) / (2m\omega_n) \quad (10)$$

である。ここで s をあらためて t とすると、(9)式は、

$$d^2x/dt^2 - f(v - dx/dt) + x = 0 \quad (11)$$

となる。これを連立一階微分方程式に直すと、

$$dx/dt = y \quad (12-1)$$

$$dy/dt = f(v \cdot y) \cdot x \quad (12-2)$$

となる。

4.2 速度摩擦特性 (12)式で示される摩擦振動の方程式は、(6)式で示される式の一つである。すなわち形の上では(6)式における非線形項の f が、速度摩擦特性になっている。ここでは、 $v \cdot y = u$ とし、 f を以下のようにモデル化して解くことにする。

$$u > 0 : f(u) = c_0 + c_1 u + c_2 / (u + c_3)$$

$$u < 0 : f(u) = -c_0 + c_1 u - c_2 / (-u + c_3)$$

$$u = 0 : f(u) = 0$$

なお、 c_0 と、 c_1 、 c_2 、 c_3 を含む各項は、物理的意味として以下の摩擦項目に対応する。

- c_0 : 乾性摩擦の項
- c_1 : 粘性摩擦の項
- $c_2 / (u + c_3)$: 境界摩擦の項

4.3 プログラム リエナールのクラスを継承してクラス定義する。

```
(defclass stick-slip ()
  ((c0 0.1)
   (c1 0.2)
   (c2 0.1)
   (c3 0.1)
   v0)
  (Lienard)
  :gettable :settable)
```

非線形項の速度摩擦特性をあらわすメソッド f を定義する。

```
(defmethod (stick-slip f)
  (let ((u (v0 - y)))
    (cond ((u > 0) (+ c0 (c1 * u) (c2 / (u + c3))))
          ((u < 0) (- c0 (c1 * u) (c2 / (-u + c3))))
          (t 0.0))))
```

煩雑と思われるスティックスリップのモデルもオブジェクト指向を用いると、このように、クラスとメソッドを一つずつ定義するだけですむ。

4.4 シミュレーション実行例 ベルトの移動速度が小さいときのシミュレーションを行なう。

(setq aMovingCase (make-instance 'stick-slip v0 0.1))
として、aMovingCaseにメッセージを送ればよい。振動波形を図10、位相面図を図11に示す。さらに速度が大きくなった場合のトラジェクトリを図12および図13に示す。

図10 ベルト移動速度が小さい時の摩擦振動波形

図11 ベルト移動速度が小さい時の位相面図

図12 ベルト移動速度が大きい時の位相面図

図12の場合はスティックスリップを生じているが、図13の場合は、減衰振動になり、スティックスリップは生じなくなる。

図13 摩擦振動が消滅する時の位相面図

5. 結果と考察

5.1 アナログ計算機との類似性 以上、オブジェクト指向プログラミングを位相面解析に適用し、各種振動系について、解法を示した。以上から分かる通り、この手法は、従来のアナログ計算機を用いてXYレコーダに出力する解析手法に似ている。すなわち、オブジェクト指向においては、クラス定義、メソッド定義、インスタンス生成、メッセージ送出といった手順で解を得るのに対し、アナログ計算機では、パッチボードの選択、パッチボード上の結線、ポテンシオメータの設定、実行という手順で解を得ていた。両者を比較すると、クラス定義とパッチボードの選択、メソッド定義とパッチボード上の結線、インスタンス生成とポテンシオメータの設定、メッセージ送出と実行といったおおまかな対応付けができる。例えば、本手法においては、インスタンス生成の際に各種パラメタを変化させてトラジェクトリへの影響を調べたが、これは、丁度、アナログ計算機のポテンシオメータの値を変更し、パラメタの影響を調べることに相当する。

また、アナログ計算機は、現実の世界のシミュレータとして用いられたが、オブジェクト指向によるプログラムモジュールは、計算機の世界におけるシミュレーション用のビルディングブロックとして用いることができる。以上のように、オブジェクト指向を用いた位相面解析手法は、煩雑な数式的な処理を行なうことなく、解を得ることができると同時に、系の大局的な性質を把握することができ、従来のアナログ計算機による解法の特長をそのまま引き継いでいると言える。

なお、アナログ計算機は、アナログであるが故に、かなり大きな誤差を生じ、精度を要求される計算には向かなかったが、本手法の場合はかなり正確な解を得ることができる。

5.2 継承機能⁽¹⁶⁾ 以上のプログラムにおける、クラス、インスタンス変数、メソッドの継承関係を示すと図14のようになる。オブジェクト指向プログラミングにより、以上述べてきた振動系は、単振動を基本クラスとして階層的に構成することが可能である。単振動のサブクラスである減衰振動系は差分となる減衰項を追加した式で表されるので、インスタンス変数として減衰係数を追加すればよい。Van-der-Polのクラスも差分となるパラメタ μ を追加し、非線形項を含む微分方程式をメソッドとして定義した。

Lienardのクラスは、減衰項が速度の関数で与えられる場合の一般的な振動系で、インスタンスを持たない抽象クラスである。Lienardのクラスの非線形減衰項の例としては、Rayleighの振動系や摩擦振動が挙げられるので、これらが更に具体的なサブクラスとして定義されている。これらのクラスは、Lienardのクラスの性質を持つ振動系であり、Lienard曲線をはじめとして、Lienardのクラスで定義された性質が全て継承されている。

図14 継承関係

5.3 プログラム量の減少 必要な機能が、その機能をサポートする最も抽象度の高いクラスで定義され、それ以下のクラスでは再利用されているので、プログラムは少ないコード量で済んでいる。例えば、Rayleighのクラスにおいて、振動波形を描く手続きを、オブジェクト指向を用いずに関数のみで定義すると、

```
(defun rayleigh-transient (dt maxtime)
  (do ((time 0 (+ dt))
      (x 1.0)
      (y 0.0)
      (mu 0.1)
      (dx dy))
      ((time > maxtime) nil)
      (setq dx (* y dt))
      (setq dy
        (* -1 (+ (* mu (- y (/ (* y y) / 3.0)) x) dt))
        (transient x y)))
```

となり、プログラム行数は、11行となるが、オブジェクト指向の場合には位相面図を描く機能をも含め、6行にすぎない。

以上述べてきた各種振動系の振動波形と位相面図を描かせることを関数のみで定義した場合と、オブジェクト指向で定義した場合についての、プログラム量の比較を表1に示す。

表1 関数とオブジェクト指向によるプログラム量の比較*

Item	Functional	Object-Oriented
simple-oscillation	18 (379)	20 (450)
dumped-oscillation	22 (483)	8 (210)
Van der Pol	22 (461)	7 (187)
Lienard	0 (0)	7 (150)
Rayleigh	22 (447)	6 (125)
stick-slip	40 (753)	13 (263)
TOTAL	124 (2523)	61 (1385)

*program lines (characters)

基本クラスに相当する部分を除き、オブジェクト指向技術の適用により、コード量を大幅に減少させることができる。なお、グラフィック機能については、関数のみでサポートするのは困難なので、メソッド、draw-fを含め比較の対象から除いている。

6. あとがき

以上、オブジェクト指向を有効に適用できる分野として非線形振動系を対象にした位相面解析法を取り上げ、具体的なプログラムを通じてその効果を述べた。

本手法によるモデル化を通じて系の大局的な性質を把握することができ、同時に、煩雑な数式処理を行なうことなく、アナログ計算機的に解を得ることができる。また、データや手続きの一元的管理、再利用を通じてプログラミングのコード量を減少させることができる。このモデル化手法は、より複雑な多自由度の非線形振動系にも適用可能である⁽¹⁷⁾。また、非線形振動系の概念を拡張した非線形制御系や生態モデル系等についても本手法は適用可能であると考えられ、さらに検討したいと考えている。

文 献

- (1) 高橋, 振動工学演習(I), (昭37), 12. オーム社
- (2) ギブソン, 非線形自動制御, (昭43), 273. コロナ社
- (3) Kerr, R.K. ほか1名, Use of Object-Oriented Programming in a Time Series Analysis System, OOPSLA '87 Proceedings, (1987), 1.
- (4) 石田好輝ほか1名, ダイナミカルモデルにおける定性的シミュレーションアルゴリズム, 計測自動制御学会論文集, 27-1, (平3), 101.
- (5) 栗田裕ほか1名, 摩擦式電磁クラッチ・ブレーキの鳴き, 機論(C), 51-467, (昭60), 1725.
- (6) Haessig Jr. D.A., ほか1名, On the Modeling and Simulation of Friction, Trans. ASME, J. Dyn. Syst. Meas. Control, 113, (1991), 354.
- (7) Streater, J.L., ほか2名, Lubricant Performance in Magnetic Thin Film Disks With Carbon Overcoat - Part 1: Dynamic and Static Friction, Trans. ASME, J. Tribol., 113, (1991), 22.
- (8) 山田康宏ほか3名, AIワークステーションELISの検討, 電子通信学会技術報告, マルチメディア通信と分散処理, 31-2, (昭61), 1.
- (9) Takeuchi, I. ほか2名, TAO - A Harmonic Mean of Lisp, Prolog, and Smalltalk, ACM SIGPLAN Notices, 18-7, (1983)
- (10) 高橋, 振動工学演習(II), (昭38), 213. オーム社
- (11) Struble, R. A., Nonlinear Differential Equations, (1962), 188. McGraw-Hill.
- (12) 高橋, 振動工学演習(II), (昭38), 225. オーム社
- (13) Struble, R. A., Nonlinear Differential Equations, (1962), 191. McGraw-Hill.
- (14) Struble, R. A., Nonlinear Differential Equations, (1962), 156. McGraw-Hill.
- (15) 曾田範宗, 摩擦の話, 岩波新書, No.791, (昭46) 167. 岩波書店
- (16) Wegner, P., The Object-Oriented Classification Paradigm, Research Directions in Object-Oriented Programming, (1987), 480, The MIT Press.
- (17) 大野, 機論, 59-568, C(平5), 64.