Jによる微分方程式の数値解とそのグラフィックス
－力学(いろいろな振動)の問題を例として－

西川　利男

１．「微分方程式は数値計算で解く」

物理学の教科書は次のように始められる[1]。力学ではニュートンの運動方程式と呼ばれる２階微分方程式を解けばどんな運動でも明らかになるという。ところが簡単な場合を除いて、解析的な数学計算だけでは解けないので、いろいろな運動の実際は明らかにされることなく、教科書は次へと進んでしまう。物理学は微分方程式を解く数学にあるのではなく物理現象を理解することにある。数値解とそのグラフィックスの役割はこの意味から極めて大きい。

このようなテーマをかかげて、すでに何回か発表を行った[2,3]。今回はこの続きとして、２階の微分方程式を力学の問題に適用しグラフィックスとして表示する課題[5]をＪで行ってみた。なお好評であったが方向場表示についてはあらためて報告する。

２．力学における微分方程式

力学の問題を解くニュートンの運動方程式は次のようである。

$$m\frac{d^2x}{dt^2} = f$$　ここでmは質量、fは力である。

数学的には、一般にこの２階微分方程式は

$$\frac{d^2x}{dt^2} + P\frac{dx}{dt} + Qx + R = 0$$

$$\begin{cases} v = \dfrac{dx}{dt} \\ \dfrac{dv}{dt} + Pv + Qx + R = 0 \end{cases}$$

のような連立１階微分方程式系と同等として計算できる。

なお、ここでは時間tに対して、変位x、速度vとする。これは一方、数学的には独立変数tに対して、従属変数xとvとなる。

---

[1] たとえば、原島鮮「力学」裳華房(1970)
[2] 西川利男「微分方程式をAPL/J思考から理解する」J研究会資料2006/9/30
[3] 西川利男「Jによる微分方程式のグラフィック・アプローチーその１
１階常微分方程式－数値解と方向場表示の活用」J研究会資料　2006/10/28
[4] 洲之内治男、寺田文行、四条忠雄「FORTRANによる演習数値計算」
サイエンス社(1980)
[5] 平田邦夫「BASICによる物理」共立出版(1988)

Numeric Solution of Differential Equation and its Graphic Display on J
- Application to Several Oscillation Problems in Dynamics -
by Toshio Nishikawa, 2010/7/24

３．微分方程式の数値解

次のようないろいろな方法がある[4]。

Euler 法 ……初期値と微分係数をもとに次々と延長して近似解を得る。
簡単だがグラフ表示ではこれで十分なことも多い。

Runge-Kutta 法 ……最も広く使われる有名な近似計算アルゴリズム

Adams-Barshforth 法…さらに良い近似を得るアルゴリズム

最も有名な Runge-Kutta 法は次のアルゴリズムからなる繰り返しで計算される。

$$x_0 = a$$

$$x_{i+1} = x_i + h, \quad h = (b - a)/n$$

としたとき

$$Y_0 = y_0$$

$$Y_{i+1} = Y_i + h(k_1 + 2k_2 + 2k_3 + k_4)/6$$

ただし

$$k_1 = f(x_i, Y_i)$$

$$k_2 = f(x_i + (1/2)h, Y_i + (1/2)hk_1)$$

$$k_3 = f(x_i + (1/2)h, Y_i + (1/2)hk_2)$$

$$k_4 = f(x_i + h, Y_i + hk_3)$$

さらに繰り返しを多段で行うには、Adams-Barshforth 法が良いとされる。
ここでは、最初の４項を Runge-Kutta 法で求め、続いて Adams-Barshforth 法で行うという組み合わせ法を採用してみた。

４．Ｊプログラミング－微分方程式を副詞で定義

プログラムをわかり易くする上で、微分方程式の計算で使用する数多くのパラメータ、引数をどうするかが重要である。とくに微分方程式という関数をＪでどう扱うかは、大きなポイントになる。筆者は微分方程式を動詞で、ルンゲクッタなどの繰り返し処理を副詞で、その他のパラメータを左右の名詞の引数とした。

左引数	動詞	副詞	右引数
(h, n)	(Diff. Equation)	srk	(X=X0, Y=Y0, Z=Z0)
きざみ, 回数	関数表示		初期値

また、数学、力学など教科書によっては、微分方程式の変数、関数の表記法に違いがあるが、これにも対応できるようにした。

数学では…(x, y, z)

物理では…(t, x, v) や (t, y, v)など

先の微分方程式の数値解を Runge-Kutta と Adams-Barshforth についてＪでプログラミングして比較してみる。Ｊのコーディングは本稿の最後にあげた。

Runge-Kutta と Adams との比較
　　- 例題……洲之内の教科書とのチェック p.126 5.2 -> p.198
　y''= - 2y*y' , x(0)=0, y(0)=0, y'(0)=1
ここでは(x, y, z)表記を用いている。また、微分方程式を J で動詞として扱うため、次のような表記を行っている。
　d（微分方程式の表す文字列）……d: 文字列を動詞に変換、初期値への設定など

```
   0.1 ((d'z');(d'0: - (2: * (y*z))')) srk^:(i.11)  0 0 1
  0         0        1
0.1 0.0996675 0.990066
0.2  0.197374 0.961043
0.3  0.291311 0.915137
0.4  0.379948 0.855639
0.5  0.462116 0.786448
0.6  0.537048 0.711578
0.7  0.604367  0.63474
0.8  0.664036 0.559055
0.9  0.716297 0.486918
  1  0.761593 0.419975
   0.1 ((d'z');(d'0: - (2: * (y*z))')) srkad  0 0 1
  0         0          1
0.1 0.0996675   0.990066
0.2  0.197374   0.961043
0.3  0.291311   0.915137
0.4  0.379909   0.855786
0.5  0.462089   0.786761
0.6  0.537052   0.712011
0.7  0.604425   0.635246
0.8  0.664167    0.55956
0.9  0.716499   0.487368
  1  0.761863   0.420334
```

５．Ｊプログラミング－'plot'によるグラフ表示

　簡単に
　　require ‘plot’
として、
　　plot X;Y
により、グラフ表示できる。
　ここで plot の右引数は

```
+-------------+-------------+
|X0 X1 X2 ... |Y0 Y1 Y2 ... |
|             |Z0 Z1 Z2 ... |
+-------------+-------------+
```

とすると、
　横軸上の X0, X1, X2, … に対して、
　縦軸上で Y0, Y1, Y2, … と Z0, Z1, Z2, …

の２本のプロット曲線が描かれる。

６．単振動と振幅の大きな振り子

［5］平田邦夫「BASICによる物理」p.31, 共立出版(1988)

長さl[m]の糸の一端におもりを吊るし、多端を固定して角度 θ から鉛直面で振らす、つまり単振り子の運動方程式は次のように与えられる。

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\sin\theta$$

ここで、振り子の振幅 θ が小さいときは$\sin\theta$ は θ に近似されこれは単振動と同じで、簡単に微分方程式は解ける。しかしながら、振幅 θ が大きいときには、この微分方程式を解析的に解くには第１種の完全楕円積分という大変やっかいな問題になる。

［1］原島鮮「力学」p.84-93, 裳華房(1970)

そこで、数値解により、グラフィックスとして解決してみよう。まず、変数表示を(t, θ, v)から(t, x, v)にする。また、きざみ間隔(h)、回数(n)、初期値(t0, x0, v0)、振り子の長さ(L)などを引数として与えるようにする。

基本となる微分方程式の数値計算部分は次のようになる。

```
(h, n) (F1;F2) odes 0, X, 0
```

ここで、微分方程式は(F1;F2)なる動詞でodesはRunge-Kutta,Adams-Barshforthの繰り返し処理の副詞である。左引数として(h, n)を、右引数として初期値(t0, x0, v0)をとる。

Ｊのプログラムは次のようになる。文字列の変換にamdstrを用いた。

```
pend_dif =: 1 : 0
:
'h n' =. x.
'X L D' =. y.        NB. X: initilal x, L: length / D=1: plot, D=0: nonplot
X =. rfd X           NB. rfd: radian from degree
'F1 F2' =. u.        NB. diff_equation
F2 =. ((":X),'"_') amdstr 'x' of F2
F2 =. ((":L),'"_') amdstr 'len' of F2
DA =. (h, n) (F1;F2) odes 0, X, 0          NB. initail t0, x0, v0
if. D = 1
  do. require 'plot'
      plot <"(1)|: 2{."(1) DA
  else.
      DA
end.
)
```

実行するには次のように行う。

```
(h, n) ((d'v');(d'-@((9.8"_%len)*(sin@x))')) pend_dif  X, L, D
```

左引数(h, n) …　きざみ間隔、繰り返し回数

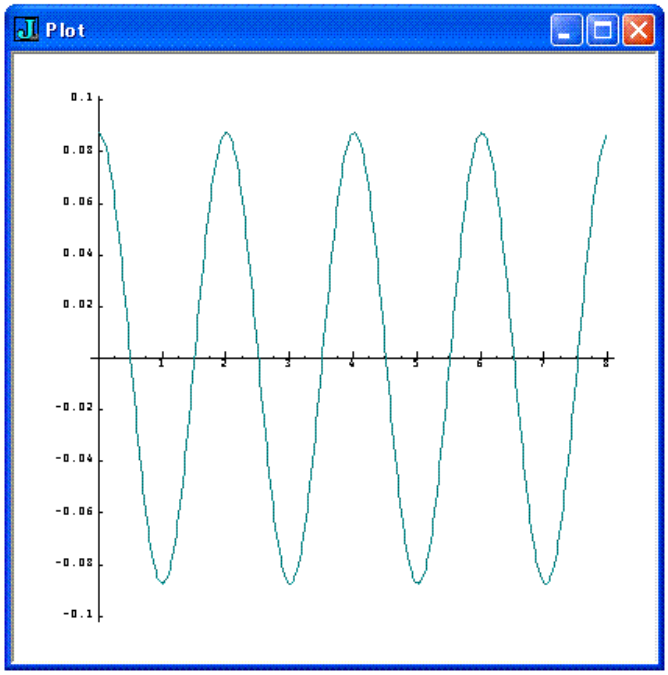
微分方程式　dv/dt = - (9.8 / l) * sin x

これにに対応するＪでの動詞の表し方に注意されたい。

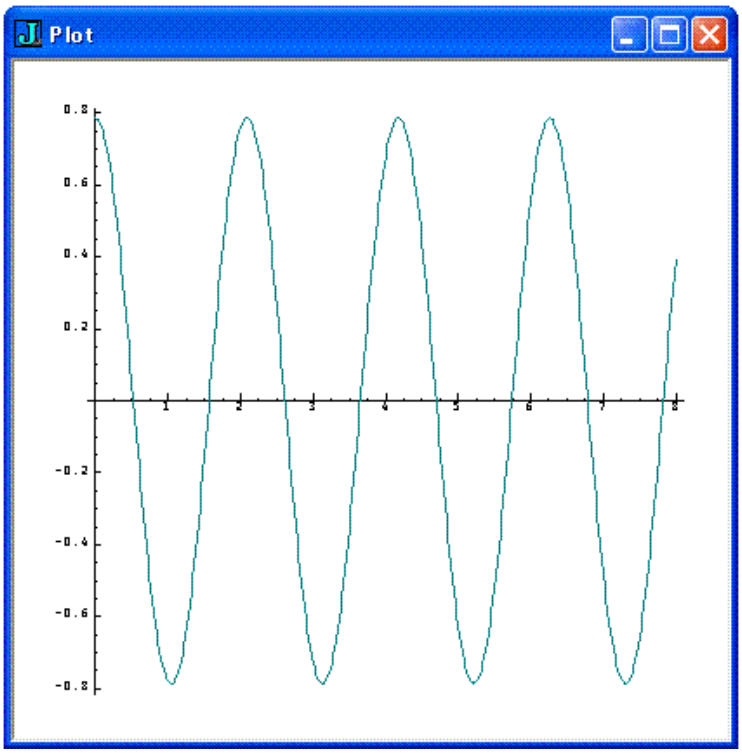
繰り返し数値計算の処理pend_dif …　副詞

右引数(X, L, D) …　初期角度(度単位), 振り子の長さ(m), プリントの可否
　　実行例は次のようになる。

```
(0.02, 401)((d'v');(d'-@((9.8"_%len) * (sin@x))')) pend_dif  5, 1, 1
```




```
(0.02, 401)((d'v');(d'-@((9.8"_%len) * (sin@x))')) pend_dif  45, 1, 1
```



これから、1ｍの振り子の初期角度が5度であれば、周期1秒できれいに等しい周期振動するが、初期角度が45度にもなると、周期は次々とずれて行くことが分かる。

７．減衰振動

［5］平田邦夫「BASICによる物理」p.36-37，共立出版(1988)

単振動には、振幅の小さいときの振り子、変位に比例した力が働くばねの振動などがある。さらに現実の力学現象には摩擦力や抵抗力が働く。つまり、速度に比例した力が働き、このとき減衰振動となる。この場合の力学をやってみよう。

運動方程式は次のようになる。

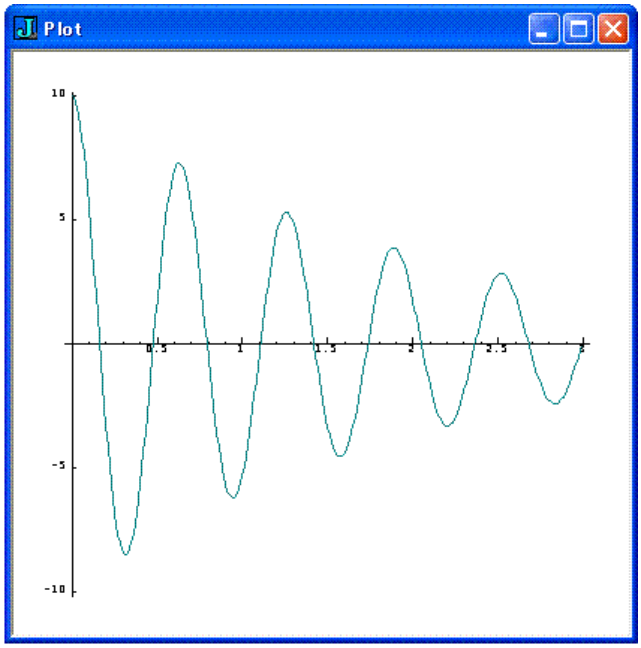
$$\frac{dv}{dt} = -w^2 x - kv$$

これに相当するJのプログラムは次のようになる。

```
damp_dif =: 1 : 0
:
'h n' =. x.
'K W D' =. y.        NB. Coef of Dif_eq. K, W
'F1 F2' =. u.
F2 =. (('':K),''_') amdstr 'k' of F2
F2 =. (('':W),''_') amdstr 'w' of F2
DA =. (h, n)(F1;F2) odes 0 10 0
if. D = 1
  do. require 'plot'
      plot <''(1)|: 2{.''(1) DA
  else.
      DA
end.
)
```

実行は例えばk=1，w=10のときは、次のように行う。

```
(0.01, 300)((d'v');(d'-@((k*v) + (w*w*x))')) damp_dif  1, 10, 1
```



減衰振動は、k と w の値の大小に応じて、３つの場合に分けられる。

k/2 < w … 不足減衰 (light damping)

k/2 = w … 臨界減衰 (critical damping)

k/2 > w … 過減衰 (heavy damping)

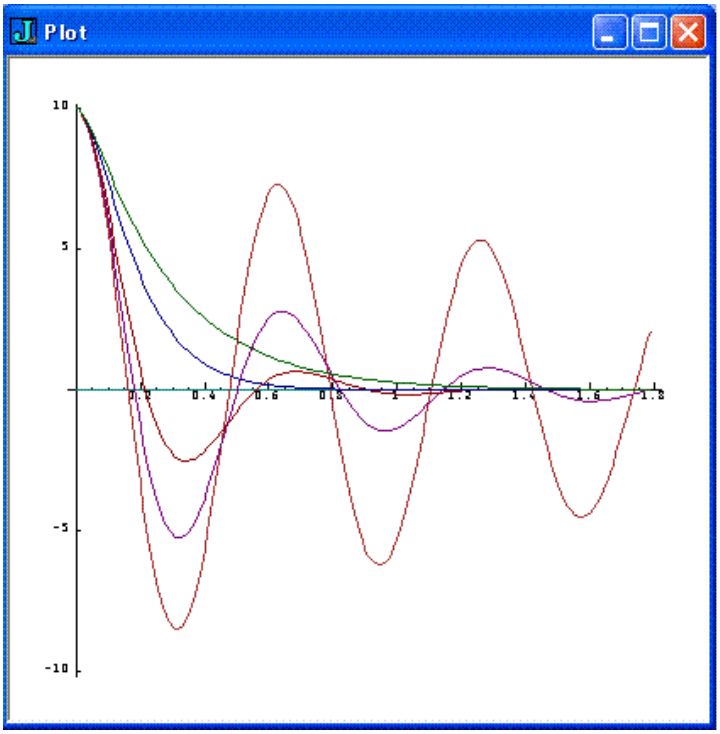
このようすを見るため、k のいろいろな値に対して比較する次のプログラムを作ってしてみた。

```
damp_osc =: 3 : 0
require 'plot'
i =. 0
Y =. ''
while. i < #y. do.
  DA =. (0.01, 180)((d'v');(d'-@((k*v) + (w*w*x))')) damp_dif (i{y.), 10, 0
  Y =. Y , ,: 1{"(1) DA
  i =. i + 1
end.
plot (0{"(1) DA); Y
)
```

実行は次のようになる。

```
damp_osc  1, 4, 8, 20, 30
```

k=1,4,8 では不足減衰、k=20 は臨界減衰、k=30 では過減衰になる。



７．強制振動

［5］平田邦夫「BASICによる物理」p.38-40，共立出版(1988)

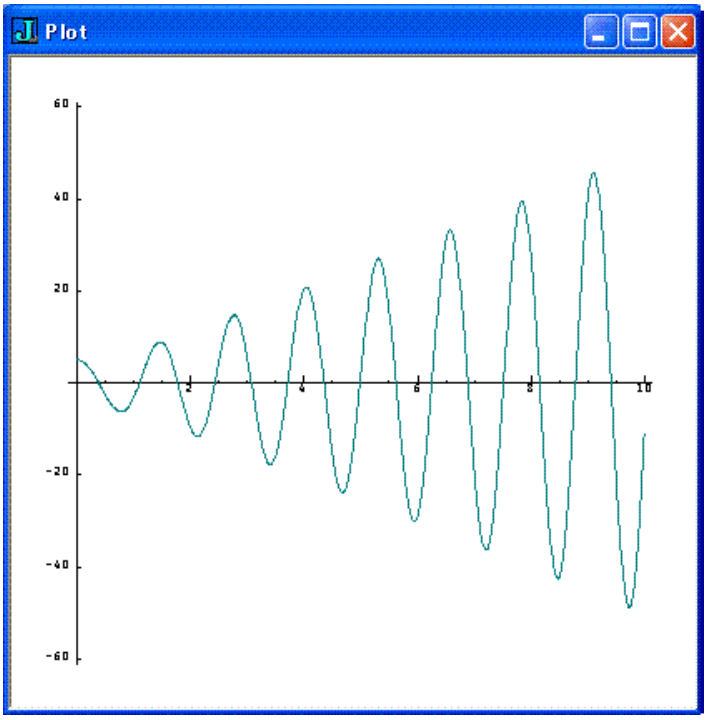
単振動に外から周期的な力が働くときの振動を強制振動という。このときの運動方程式とＪのプログラムは次のようになる。

$$\frac{dv}{dt} = -w^2x - kv - p\cos qt$$

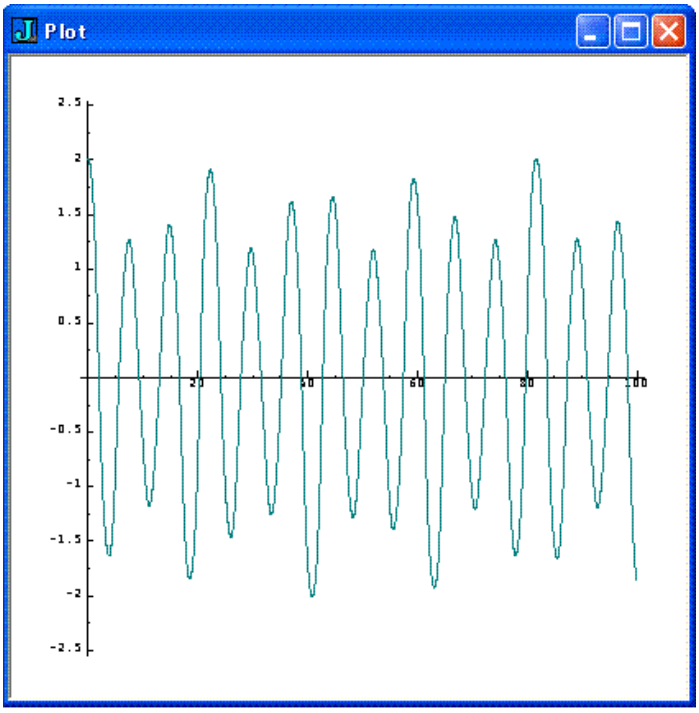
```
forced_dif =: 3 : 0
:
'K P Q W D' =. y.  NB. K, W, P, Q: Coef of Dif_eq.
F2 =. (('':K),'''_') amdstr 'k' of F2
F2 =. (('':P),'''_') amdstr 'p' of F2
F2 =. (('':Q),'''_') amdstr 'q' of F2
F2 =. (('':W),'''_') amdstr 'w' of F2
DA =. (0.01, 1000) (F1;F2) odes 0 5 0
if. D = 1
  do. require 'plot'
      plot <''(1)|: 2{.''(1) DA
  else.
      DA
end.
)
DIF_EQ1 =: '(-@((k*v)+(w*w*x)))'
DIF_EQ2 =: '(p * cos@(q*t))'
DIF_EQ  =: DIF_EQ1, '+', DIF_EQ2
```

微分方程式はこのように分割しても入れられる。

```
((d'v');(d DIF_EQ)) forced_dif 0, 50, 5, 5, 1
```



```
((d'v');(d DIF_EQ)) forced_dif 0, 70, 4, 5, 1
```



共振の場合（w=5, q=5）には振幅がしだいに大きくなる。一方、共振していない場合（w=5, q=4）には振幅は大きくなったり、小さくなったりする。

８．パラメータ励振

［5］平田邦夫「BASICによる物理」p.41-42, 共立出版(1988)

単振り子の場合で糸の長さlが時間とともに変化するときを考えてみる。ぶらんこをこぐとき、身体の重心を上下させることで振幅がしだいに大きくなる。これは振り子の長さを時間的に変化させることに対応する。このように長さのようなパラメータが変化することで振動が成長する現象をパラメータ励振(parametric excitation)といい、非線形の振動になる。この場合の運動方程式は次のようになる。

$$\frac{d^2\theta}{dt^2}+\frac{2}{l}\frac{dl}{dt}\frac{d\theta}{dt}+\frac{g}{l}\sin\theta = 0$$

$$l = l_0\{1+b\cos(wt+p)\}$$

$\theta = x,\quad \frac{d\theta}{dt} = v$ とおきかえると

$$\frac{dv}{dt} = \frac{2bw\sin wt\cdot v - w^2\sin x}{1+b\cos(wt+p)}$$

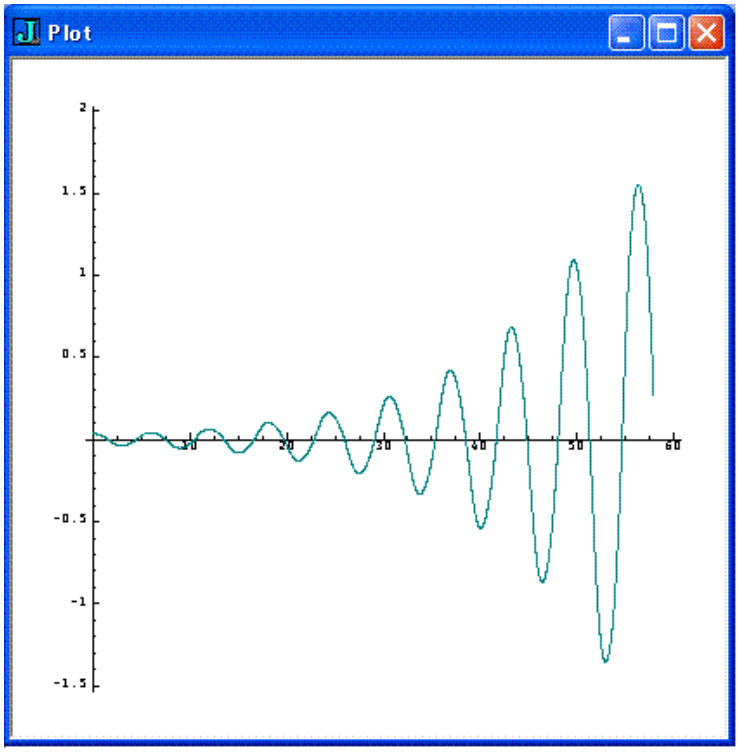
当然のように、解析的な解はMathieu方程式と呼ばれる微分方程式になり、ふつうには解けない。数値計算のありがたみがわかる。

これに対応するJのプログラムは次のようになる。

```
NB. Parametric Excited Oscillation
PARAM1 =: '(v*(2:*b)*w*sin@(w*t))'
PARAM2 =: '((r*r)*sin@x)'
PARAM12 =: PARAM1, '-', PARAM2
PARAM3 =: '(1:+b*cos@(p+w*t))'
PARAM  =: '(', PARAM12, ') % ',PARAM3
param_dif =: 3 : 0
:
'B R P X W D' =. y.  NB. D=1 plot, D=0 return DA
X =. rfd X
'F1 F2' =: x.
F2 =. ((":B),'"_') amdstr 'b' of F2
F2 =. ((":R),'"_') amdstr 'r' of F2
F2 =. ((":P),'"_') amdstr 'p' of F2
F2 =. ((":W),'"_') amdstr 'w' of F2
DA =. (0.01, 5800) (F1;F2) odes 0, X, 0
if. D = 1
  do. require 'plot'
      plot <"(1)|: 2{."(1) DA
  else.
      DA
end.
)
```

次のように実行する。振幅は最初の 2° からしだいに大きくなる。

```
  ( (d'v');(d PARAM) ) param_dif 0.1 1 0 2 2 1
```



```
NB. 平田邦夫「BASICによる物理」p.37-40 ================================

NB. Pendulum comapared by  ================================================

NB. (0.02, 401)((d'v');(d'-@((9.8"_/len) * (sin@x))')) pend_dif  45, 1, 1 NB.
45deg, 1m
NB. (0.02, 401)((d'v');(d'-@((9.8"_/len) * (sin@x))')) pend_dif  5, 1, 1  NB.
5deg, 1m
pend_dif =: 1 : 0
:
'h n' =. x.
'X L D' =. y.        NB. Coef of Dif_eq. K, W
X =. rfd X
'F1 F2' =. u.
F2 =. ((":X),'"_') amdstr 'x' of F2
F2 =. ((":L),'"_') amdstr 'len' of F2
DA =. (h, n)(F1;F2) odes 0, X, 0 NB. initail t0, x0, v0
if. D = 1
  do. require 'plot'
      plot <"(1)|: 2{."(1) DA
  else.
      DA
end.
)

NB. Damped Oscillation p.37 ===============================================
NB. damp_osc  1, 4, 8, 20, 30
NB. damp_osc  1  => k/2<w(10) light damp oscillation
NB. damp_osc 20  => k/2=w(10) critical damp oscillation
NB. damp_osc 30  => k/2>w(10) heavy damp oscillation

damp_osc =: 3 : 0
require 'plot'
i =. 0
Y =. ''
while. i < #y. do.
  DA =. (0.01, 180)((d'v');(d'-@((k*v) + (w*w*x))')) damp_dif (i{y.), 10, 0
  Y =. Y , ,: 1{"(1) DA
  i =. i + 1
end.
plot (0{"(1) DA); Y
)

NB. dv/dt + k*v + w*w*x = 0 : (t, x, v)
NB.                                          K  W   D
```

```
NB. (0.01, 180)((d'v');(d'-@((k*v) + (w*w*x))')) damp_dif  1, 10, 1  NB.
light damp
NB. (0.01, 180)((d'v');(d'-@((k*v) + (w*w*x))')) damp_dif  2, 10, 1  NB.
light damp
NB. (0.01, 180)((d'v');(d'-@((k*v) + (w*w*x))')) damp_dif 20, 10, 1  NB.
critical damp
NB. (0.01, 180)((d'v');(d'-@((k*v) + (w*w*x))')) damp_dif 30, 10, 1  NB.
heavy damp

NB. using convert routine d
NB. adverb definition / modified 2010-6-30
NB. (0.01, 180) ((d'v');(d'-@((k*v) + (w*w*x))')) damp_dif  2, 10, 1  NB.
light damp
damp_dif =: 1 : 0
:
'h n' =. x.
'K W D' =. y.        NB. Coef of Dif_eq. K, W
'F1 F2' =. u.
F2 =. ((":K),'"_') amdstr 'k' of F2
F2 =. ((":W),'"_') amdstr 'w' of F2
NB. DA =. (0.01, 180) (F1;F2) odes 0 10 0
DA =. (h, n)(F1;F2) odes 0 10 0
if. D = 1
  do. require 'plot'
      plot <"(1)|: 2{."(1) DA
  else.
      DA
end.
)

NB. Forced Oscillation p.39 ========================================
NB. forced_osc 5, 30
forced_osc =: 3 : 0
require 'plot'
i =. 0
NB. Y =. 0 0$''
Y =. ''
while. i < #y. do.
  DA =. ( (d'v');(d'(-@((k*v)+(w*x))) + (p * cos@(q*t))') ) forced_dif 1, 50,
(i{y.), 1000, 0
  Y =. Y , ,: 1{"(1) DA
  i =. i + 1
end.
plot (0{"(1) DA); Y
)
```

```
NB. make dif_eq. of parts
=====================================================

DAMP1 =: '(-@((k*v)+(w*x)))'
DAMP2 =: '(p * cos@(q*t))'
DAMP  =: DAMP1, '+', DAMP2

NB. ((d'v');(d DAMP)) forced_dif 1, 50, 20, 1000, 1

NB. ( (d'v');(d'(-@((k*v)+(w*x))) + (p * cos@(q*t))') ) forced_dif 1, 50, 20,
1000, 1
forced_dif =: 3 : 0
:
'K P Q W D' =. y.  NB. K, W, P, Q: Coef of Dif_eq. / D=1 plot, D=0 return DA
'F1 F2' =. x.
F2 =. ((":K),'"_') amdstr 'k' of F2
F2 =. ((":P),'"_') amdstr 'p' of F2
F2 =. ((":Q),'"_') amdstr 'q' of F2
F2 =. ((":W),'"_') amdstr 'w' of F2
DA =. (0.01, 1440) (F1;F2) odes 0 5 0
if. D = 1
  do. require 'plot'
      plot <"(1)|: 2{."(1) DA
  else.
      DA
end.
)

NB. Parametric Excitation p. 41
=====================================================

PARAM1 =: '(v*(2:*b)*w*sin@(w*t))'
PARAM2 =: '(r*sin@x)'
PARAM12 =: PARAM1, '-', PARAM2
PARAM3 =: '(1:+b*cos@(p+w*t))'
PARAM  =: '(', PARAM12, ') % ',PARAM3
NB. ( (d'v');(d PARAM) ) param_dif 0.1 1 0 1 2 1

NB. ( (d'v');(d'((v*(2:*b)*w*sin@(w*t))) - (r*sin@x)) %
(1:+b*cos@(p+w*t))') ) param_dif 0.1 2 0 1 2 1
param_dif =: 3 : 0
:
'B R P Q W D' =. y.  NB. D=1 plot, D=0 return DA
'F1 F2' =: x.
```

```
F2 =. ((":B),'"_') amdstr 'b' of F2
F2 =. ((":R),'"_') amdstr 'r' of F2
F2 =. ((":P),'"_') amdstr 'p' of F2
F2 =. ((":Q),'"_') amdstr 'q' of F2
F2 =. ((":W),'"_') amdstr 'w' of F2
NB. DA =. (0.01, 1440) (F1;F2) odes 0 5 0
DA =. (0.01, 10000) (F1;F2) odes 0 2 0
if. D = 1
  do. require 'plot'
      plot <"(1)|: 2{."(1) DA
  else.
      DA
end.
)

NB.
================================================================================

NB. (h, n) ((d'z');(d'dif_eq')) odes init x, y, z
NB. (0.1, 10) ((d'z');(d'0: - (2: * (y * z))')) odes 0 0 1 => (x, y, z)
NB. (0.1, 10) ((d'v');(d'0: - (2: * (y * v))')) odes 0 0 1 => (t, y, v)
odes =: 1 : 0
:
'h n' =. x.
NB. if. '' = n do. n =. 10 end.
'x y z' =. y.
srg =. h u. srk^:(i.4) x, y, z
sad0 =. , |. }."(1)  srg
sad =. h u. sadam^:(i.n-3) (x+3*h), sad0
(}: srg), 3 {."(1) sad
)

NB.
================================================================================

NB. Revised Version - Diff_Eq. as Explicit Left Argument
NB.   using convert variables function d
NB.  2006/9/17-18  Runge_Kutta and Adams4 Method
NB. 洲之内「演習数値計算」とのチェック
NB. 0.1 ((d'1: % (2: * %:@ z)');(d'0: - %@(*:@ y)')) srkad 0 1 1 p126 問題
5.1(1), 解答p.196
NB. 0.1 ((d'x + *:@ z');(d'y - x')) srkad 0 0 1                  p126 問題
5.1(2), 解答p.197
NB. 0.1 ((d'z');(d'0: - (2: * (y * z))')) srkad 0 0 1            p126 問題
5.2(1), 解答p.198
```

```
NB. 0.1 ((d'z');(d'x + (*:@y) - (2: * z)')) srkad 0 1 1          p126 問題
5.2(2), 解答 p.199
srkad =: 1 : 0
:
h =. x.
'x y z' =. y.
srg =. h u. srk^:(i.4) x, y, z
sad0 =. , |. }."(1)  srg
sad =. h u. sadam^:(i.28) (x+3*h), sad0
(}: srg), 3 {."(1) sad
)

NB. 0.1 ((d'1: % (2: * %: z)');(d'0: - %@(*:@ y)')) srk  0 1 1
srk =: 1 : 0
:
h =. x.
x =. 0{y.
Y1 =. 1{y.
Y2 =. 2{y.
F =: u.
'k11 k21' =. ". > F ,L:0 'x, Y1, Y2'
'k12 k22' =. ". > F ,L:0 '(x + h%2), (Y1 + h*k11%2), (Y2 + h*k21%2)'
'k13 k23' =. ". > F ,L:0 '(x + h%2), (Y1 + h*k12%2), (Y2 + h*k22%2)'
'k14 k24' =. ". > F ,L:0 '(x + h), (Y1 + h*k13), (Y2 + h*k23)'
(x+h), (Y1+h*(k11+(2*k12)+(2*k13)+k14)%6), (Y2+h*(k21+(2*k22)+(2*k23)+k24)%6)
)

NB. 0.1 ((d'1: % (2: * %: z)');(d'0: - %@(*:@ y)')) sadam 0.3, SC
sadam =: 1 : 0
:
h    =. x.
x    =. 0{y.
'Yi Yj'     =. 1 2{y.
'Yi_1 Yj_1' =. 3 4{y.
'Yi_2 Yj_2' =. 5 6{y.
'Yi_3 Yj_3' =. 7 8{y.
F =. u.
'Fi Fj' =. 55* ".> F ,L:0 'x, Yi, Yj'
'Fi_1 Fj_1' =. _59* ".> F ,L:0 '(x-h), Yi_1, Yj_1'
'Fi_2 Fj_2' =. 37* ".> F ,L:0 '(x-2*h), Yi_2, Yj_2'
'Fi_3 Fj_3' =. _9* ".> F ,L:0 '(x-3*h), Yi_3, Yj_3'
Yii  =. Yi + (h%24)*(Fi + Fi_1 + Fi_2 + Fi_3)
Yjj  =. Yj + (h%24)*(Fj + Fj_1 + Fj_2 + Fj_3)
(x+h), Yii, Yjj, Yi, Yj, Yi_1, Yj_1, Yi_2, Yj_2
)
```

```
NB. Covert Variables x, y, z => (0&{), (1&{), (2&{) ======================
d =: 3 : 0
f =. y.
if. 'z' e. f
  do.    NB. using (x, y, z)
    f =. '(0&{)' amdstr 'x' of f
    f =. '(1&{)' amdstr 'y' of f
    f =. '(2&{)' amdstr 'z' of f
  else.  NB. using (t, x, v) or (t, y, v)
    f =. '(0&{)' amdstr 't' of f
    f =. '(1&{)' amdstr 'x' of f
    f =. '(1&{)' amdstr 'y' of f
    f =. '(2&{)' amdstr 'v' of f
end.
f =. '(', f, ')'
)

NB. Amend in String Program by T. Nishikawa ================================
str2box=: 3 : 0
:
Z1=. x. E. y.
Z2=. (|. x.) E. (|. y.)
Z3=. 1, }: |. Z2
Z4=. Z1 +. Z3                          NB. revised to (+.) 2006/6/28
Z5=. Z4 <;.1 y.
AM=: ((<,x.) = Z5) # i. #Z5 NB. AM is global value
Z5
)

of=: str2box  NB. alias for string amend

NB. Usage f.g.
NB.    'pq' amdstr 'xyz' of 'abcxyzdefgxyzhi'
NB. abcpqdefgpqhi
NB. revised 2006/6/28, 7/3
NB.   'pq' amdstr 'xyz' of 'xyzdefgxyzhi'
NB. pqdefgpqhi
NB.   'pq' amdstr 'x' of 'xyzdefgxyzhi'
NB. pqyzdefgpqyzhi

amdstr=: 3 : 0
```

```
:
; (<,x.) AM } y.                NB. revised 2006/7/3
)
```