7

計算機の応用

運動方程式を数値的に解くためには離散化が必要である．しかし，離散化には注意が必要である．ここではまず離散化の仕組みを学ぶ．最終的には４次のルンゲ・クッタ法を使って例題を解き，数値計算の面白さを知ろう．印刷教材にＣプログラムを例示し，パソコンを所有する学生には演習問題を提供する．

1.　数値計算の重要性と計算機言語

■　数値計算の重要性　数値計算がなぜ重要なのだろうか．第１に，自然科学や工学では，方程式の解の存在証明とか解の増減といった挙動だけでは不十分で，法則の実験的検証や法則の実際的応用には具体的な数値的計算が必要不可欠だからである．第２に，解析的解法のない非線形方程式や相互作用のある多体系，本質的に挙動の予測不可能なカオスといった現象には数値的方法は最も強力な研究方法だからである．

■　計算機利用の大衆化　ハードウエアの急激な発展で，個人のパーソナルコンピューターの計算能力が，一昔前専門家が大型計算機で所有していた計算能力を凌駕するようになった．最先端がさらに新しい世界に踏み込んでいることの影響も重要だが，それ以上に，計算機利用の大衆化，裾野の拡がりはわれわれの時代を大きく変えつつある．誰もが学習する理由がここにある．

■　計算機言語　計算機，計算機言語の歴史は極めて浅い．計算機の進歩はいうまでもないが，計算機言語も進歩する．概念的に面目を一新して進歩，変容する余地がある．

しかし，計算機の数値計算的使用に限れば言語間の翻訳は容易である．なぜなら，加減乗除は普遍的概念であるから，それらの単純な組み合わせ程度では，どの言語でもプログラムの骨格は変わらないからである．例えば，Basic，Fortran，Pascal，C，C++，数式処理言語など相互の翻訳が可能である．似た単語と同様なる構造を用いるヨーロッパ言語間の翻訳程度の違いであり想像可能である．したがって，生涯教育の時代，過去に囚われず常に新しい言語に取り組む意欲が必要である．章末に参考のためにCプログラムを付けたが，他言語習得者も柔軟に対応していただきたい．

数値計算の学習で大切なのは，個々の言語の学習ではなく，言語に依らぬアルゴリズムの理解と実際の計算演習である．実践的な外国語習得における話したい内容，書きたい内容がアルゴリズムにあたり，演習は実地の会話と作文に対応すると考えてみればよい．文法の学習は付随するものである．そういった観点から，本章では微分方程式の数値解法のアルゴリズムを解説し，次に演習問題として応用例を述べた．アルゴリズムを理解し，自ら得意な言語でプログラムを書けばよい．

2.　微分方程式の数値解法の基礎概念

■　離散化　関数$y = f(x)$のxが 0 から 10 までのグラフを描くことを考えよう．このとき，0 から 10 までのすべてのxの値を計算することはできない．また，ディスプレイの表示画素より細かい計算は無意味である．例えば 0.1 刻みに計算すれば 101 回$f(x)$を計算することになる．このように，コンピューターの計算には，常に「刻み」，すなわち離散化が必要である．

さらに重要なことは，物理学の基本方程式も離散化されることである．その際，離散化の仕方は一通りではないことを注意しよう．例えば，生態系の個体数Nの時間変化を与える方程式であるロジスティック方程式

$$\frac{\mathrm{d}N}{\mathrm{d}t} = \alpha N(1 - \beta N)$$

を

$$N_{n+1} - N_n = \alpha h N_{n+1}(1 - \beta N_n)$$
$$N_{n+1} - N_n = \alpha h N_n (1 - \beta N_n)$$

のどちらに離散化するかで，そのふるまいは大きく異なる．前者はもとの方程式と同じふるまいを示すが，後者はカオスを与える写像として有名なロジスティック写像である．

「計算機にどのように乗せるか，どのように効率よく計算するか」という研究から新しい理論が生まれて物理理論の本質的理解が深まることも多い．連続的か，離散的かという問題は，物理科学のいろいろな場面で遭遇する基本的問題でもある．

■　単振動　単振動の方程式を考えよう．運動方程式は

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -kx \tag{1}$$

だから，$k/m = \omega^2$，$\omega t = s$ とおくならば，

$$\frac{\mathrm{d}^2x}{\mathrm{d}s^2} = -x \tag{2}$$

と書ける．一階連立微分方程式に直すと，

$$\frac{\mathrm{d}x}{\mathrm{d}s} = v, \quad \frac{\mathrm{d}v}{\mathrm{d}s} = -x \tag{3}$$

となる．解はA，φを定数として

$$x = A\sin(s + \varphi), \quad v = A\cos(s + \varphi) \tag{4}$$

である．

■　微分の定義による離散化　微分の定義

$$\frac{\mathrm{d}x}{\mathrm{d}s} = \lim_{h \to 0}\frac{x(s + h) - x(s)}{h} \tag{5}$$

において，計算機で極限はとれないから有限のhを用いる．これが離散化である．今，時間間隔をhで刻み，$s_n = nh$，$x(s_n) = x_n$，$v(s_n) = v_n$として(3)を離散化するならば，微分方程式は定差方程式

$$\left.\begin{aligned} x_{n+1} - x_n &= v_n h \\ v_{n+1} - v_n &= -x_n h \end{aligned}\right\} (n = 0,1,2,\cdots) \tag{6}$$

となる．この式は時々刻々の変化の割合を計算しながら，次々に x_{n+1}，v_{n+1} を求める方法を与える．

$x_0 = 1.0$，$v_0 = 1.0$ として計算してみよう．このとき，厳密解

$$x_e = \sqrt{2}\sin(s + \pi/4), \quad v_e = \sqrt{2}\cos(s + \pi/4) \tag{7}$$

と比べる．test1.c にサンプルプログラムを与えた．

表 7-1 に $h = 0.1$ での x の計算結果を示した．(6)はおおよその値を再現しているが，ステップを増すごとに誤差 $(x - x_e)/x_e$ が増加している．刻みが $h = 0.1$ では大きすぎるのだろうか？

表 7-1　計算結果

s	x	x_e（厳密）	誤差(%)	$x^2 + v^2$
0.000	1.000	1.000	0.000	2.000
0.100	1.100	1.095	0.472	2.020
0.200	1.190	1.179	0.956	2.040
0.300	1.269	1.251	1.450	2.061
0.400	1.336	1.310	1.955	2.081
0.500	1.391	1.357	2.469	2.102

■　アルゴリズムの検討　数値計算では，厳密に成り立つべき関係などを用いて，計算が正しいかチェックする必要がある．この点は，数値計算において特に注意すべき点である．

$x^2 + v^2$ を s で微分して(3)を利用すれば「$x^2 + v^2$ ＝一定」がわかる．これはエネルギー保存則である．ところが(6)から求めた値は，表 7-1 に示したように，徐々に増加する．

この理由を調べるために，１ステップごとの変化を(6)から計算すると

$$\begin{aligned} x_{n+1}{}^2 &= (x_n + v_n h)^2 = x_n{}^2 + 2x_n v_n h + v_n{}^2 h^2 \\ v_{n+1}{}^2 &= (v_n - x_n h)^2 = v_n{}^2 - 2x_n v_n h + x_n{}^2 h^2 \end{aligned} \tag{8}$$

である．したがって，

$$x_{n+1}{}^2 + v_{n+1}{}^2 = (1 + h^2)(x_n{}^2 + v_n{}^2) > x_n{}^2 + v_n{}^2 \tag{9}$$

を得る．すなわち，どれほど小さな刻みを用いても各ステップごとに必ず全エネルギーが増加することがわかる．したがって，このアルゴリズムは使えない．刻みによる誤差というよりも，計算は誤りである．

以上，微分の定義をそのままおきかえるだけの離散化では正確な計算を与えないことを見た．この例は離散化には注意が必要であることを教える．

■　改善策　上に述べた問題を改善する工夫として，次の方法がある．正確に言えば，(6)の左辺はステップnと$n+1$の中間に対応し，右辺はnだけに対応する．右辺も両ステップに対応したら良いのではないだろうか．そこで

$$\left.\begin{aligned} x_{n+1}-x_n &= \frac{1}{2}(v_{n+1}+v_n)h \\ v_{n+1}-v_n &= -\frac{1}{2}(x_{n+1}+x_n)h \end{aligned}\right\}(n=0,1,2,\cdots) \qquad (10)$$

のように右辺で両ステップの値の平均をとる．

この場合，(10)の２式を「たすき掛け計算」（上の左辺×下の右辺＝下の左辺×上の右辺）することにより，

$$x_{n+1}{}^2+v_{n+1}{}^2=x_n{}^2+v_n{}^2 \qquad (11)$$

を満たすことがわかる．

x_0	，$v_{1/2}$	$\rightarrow$	x_1
$v_{1/2}$	，x_1	$\rightarrow$	$v_{3/2}$
x_1	，$v_{3/2}$	$\rightarrow$	x_2
$v_{3/2}$	，x_2	$\rightarrow$	$v_{5/2}$

■　かえる飛び法　実際的には(10)と同じ考え方としてかえる飛び法と呼ばれる

$$\left.\begin{aligned} x_{n+1} &= x_n+v_{n+1/2}h \\ v_{n+3/2} &= v_{n+1/2}-x_{n+1}h \end{aligned}\right\}(n=0,1,2,\cdots) \qquad (12)$$

で計算すると良い．これを用いるためには，初期値として$v_{1/2}=v_0-x_0h/2$を求めておく必要がある．その後，(12)式を交互に使って囲みのように値を求めてゆくことができる．test2.c にサンプルプログラムを与えた．

エネルギー保存則をみるために，飛び越して計算していないところは隣合う２つの値を平均することにすると，表 7-2 からわかるように，値は上下しながらも，（100 ステップまで）保存則は満たされていることがわかる．

s	x^2+v^2
0.000	2.000
1.000	2.002
2.000	1.998
3.000	1.999
4.000	2.002
5.000	1.999
6.000	1.999
7.000	2.002
8.000	1.999
9.000	1.998
10.000	2.002

表 7-2　計算の検討

かえる飛び法では，初期条件として，x_0，v_0 だけでは不十分であるが，一度必要なデータを集めてしまえば，以後はデータを捨てることなく使うので，効率的な方法である．また，微分の定義を用い，時々刻々積分している感じが掴？める点でわかりやすい．

3. ルンゲ・クッタ法

■　ルンゲ･クッタ法　種々の微分方程式を与えられた初期条件だけから積分する一般的方法としてルンゲ・クッタ法がある．以下，その原理を学ぼう．

この方法は一般に微分方程式

$$\dot{x} = f(x,t) \tag{13}$$

を考える．初期条件として$t = t_0$で，$x(t_0) = x_0$とする．求めたい$x(t_0 + h)$は

$$x(t_0 + h) = x_0 + h\dot{x}(t_0) + \frac{1}{2}h^2\ddot{x}(t_0) + \cdots\cdots \tag{14}$$

とテイラー展開される．

$$\ddot{x} = \frac{\partial f}{\partial x}\dot{x} + \frac{\partial f}{\partial t} = f\frac{\partial f}{\partial x} + \frac{\partial f}{\partial t} \tag{15}$$

となるから，(14)は

$$x(t_0 + h) = x(t_0) + hf + \frac{h^2}{2}\left(f\frac{\partial f}{\partial x} + \frac{\partial f}{\partial t}\right) + \cdots\cdots \tag{16}$$

と書ける．ただし，$f, \partial f/\partial x, \partial f/\partial x$は$t = t_0$の値である．

これと同等のものを計算するために

$$\begin{aligned} k_1 &= hf(x_0, t_0) \\ k_2 &= hf(x_0 + \alpha k_1, t_0 + h) \\ x(t_0 + h) &= x(t_0) + \beta_1 k_1 + \beta_2 k_2 \end{aligned} \tag{17}$$

とおいて，$x(t_0 + h)$ の値が h の２次の項まで (16) と一致するように α, β_1, β_2 を定めよう．(17) の２番目の式を

$$k_2 = h\left[f(x_0, t_0) + \alpha k_1 \frac{\partial f}{\partial x} + h \frac{\partial f}{\partial t} + \cdots\cdots \right] \tag{18}$$

のように展開して，３番目の式に代入すると

$$x(t_0 + h) = x(t_0) + h\left[(\beta_1 + \beta_2) f + h\left(\alpha\beta_2 f \frac{\partial f}{\partial x} + \beta_2 \frac{\partial f}{\partial t} \right) \right] \tag{19}$$

を得る．(16) と (19) を見比べて，$f, \partial f/\partial x, \partial f/\partial t$ の関数形に関係なく等しくなるためには

$$\beta_1 + \beta_2 = 1, \quad \alpha\beta_2 = \frac{1}{2}, \quad \beta_2 = \frac{1}{2} \tag{20}$$

の条件が必要である．したがって

$$\alpha = 1, \quad \beta_1 = \beta_2 = \frac{1}{2}, \tag{21}$$

すなわち，(17) は

$$\begin{aligned} k_1 &= hf(x_0, t_0) \\ k_2 &= hf(x_0 + k_1, t_0 + h) \\ x(t_0 + h) &= x(t_0) + \frac{1}{2}(k_1 + k_2) \end{aligned} \tag{22}$$

となる．(22) は h の２次までテイラー級数と一致するので，２次のルンゲ・クッタ法とよばれる．前節の (6) は $\beta_1 = 1, \ \beta_2 = 0$ に相当し，(10) は (22) に対応していることを注意しておこう．

■　汎用ルンゲ・クッタ法　計算機の手間と精度，計算機のもつ実数の桁数から考えて，よく用いられるのは以下の４次のルンゲ・クッタ法である．

$$
\begin{aligned}
k_1 &= hf(x_0, t_0) \\
k_2 &= hf(x_0 + k_1/2, t_0 + h/2) \\
k_3 &= hf(x_0 + k_2/2, t_0 + h/2) \\
k_4 &= hf(x_0 + k_3, t_0 + h) \\
x(t_0 + h) &= x(t_0) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
\tag{23}
$$

ニュートンの運動方程式は２階微分方程式

$$\ddot{x} = F(x, \dot{x}, t) \tag{24}$$

であって，前節のように１階連立微分方程式

$$\dot{x} = v, \quad \dot{v} = F(x, v, t) \tag{25}$$

と書ける．一般的には，

$$\dot{x} = f_1(x, v, t), \quad \dot{v} = f_2(x, v, t) \tag{26}$$

の形である．このとき，４次のルンゲ・クッタ法の公式は

$$
\begin{aligned}
k_1 &= hf_1(x_0, v_0, t_0) \\
m_1 &= hf_2(x_0, v_0, t_0) \\
k_2 &= hf_1(x_0 + k_1/2, v_0 + m_1/2, t_0 + h/2) \\
m_2 &= hf_2(x_0 + k_1/2, v_0 + m_1/2, t_0 + h/2) \\
k_3 &= hf_1(x_0 + k_2/2, v_0 + m_2/2, t_0 + h/2) \\
m_3 &= hf_2(x_0 + k_2/2, v_0 + m_2/2, t_0 + h/2) \\
k_4 &= hf_1(x_0 + k_3, v_0 + m_3, t_0 + h) \\
m_4 &= hf_2(x_0 + k_3, v_0 + m_3, t_0 + h) \\
x(t_0 + h) &= x(t_0) + (k_1 + 2k_2 + 2k_3 + k_4)/6 \\
v(t_0 + h) &= v(t_0) + (m_1 + 2m_2 + 2m_3 + m_4)/6
\end{aligned}
\tag{27}
$$

で与えられる．

(27)を用いて，どんな問題にも使える汎用プログラムを作ることができる．fだけ問題ごとに変えればよいのである．

4.　一次元の運動

■　単振り子　単振り子の運動方程式

$$\ddot{\varphi} = -\frac{g}{l}\sin\varphi \tag{28}$$

を考える．ただし，ここでgは重力加速度9.8[m/s^2]，lは振り子の長さである．この方程式は簡単な関数で解くことができないので，数値的に解いてみよう．

簡単のために$\omega^2 = \dfrac{g}{l}$，$\omega t = s$とおくことによって

$$\frac{\mathrm{d}^2\varphi}{\mathrm{d}s^2} = -\sin\varphi \qquad (29)$$

となる．$l = 25$［cm］としてこの運動の様子と周期を求めてみよう．test3.cでは最初にφが負になった時から次に正になるまでの時間で周期を測定した．

図 7-1(a)に振幅を規格化した振動の様子をプロットした．明らかに曲線は一致しない．振動が小さいうちは三角関数曲線と見なせるが，最大振幅 175 度の曲線では，振幅の大きい部分がふくらみ三角関数曲線とは言えない．

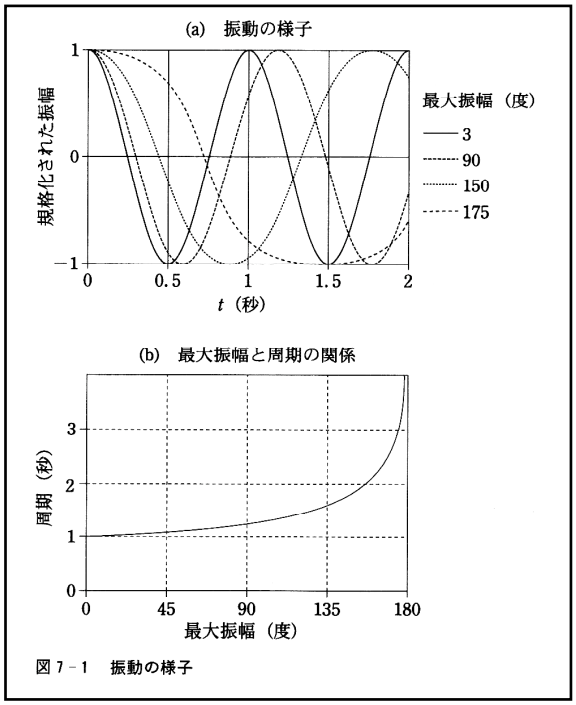

図 7－1　振動の様子

図 7-1(b)に最大振幅と周期の関係を示した．ある程度大きな振れまではそれほど周期は増加しない．この性質はガリレオによって発見された等時性とよばれる性質である．確かに振り子が建物の天井からつるされた非常に長いものであるならば，その振れ角は微小と見なされ，等時性が成立する．すなわち，振幅によらず周期が一定であるという単振動と考えてよかった．

ところが図 7-1(b)に見られるように，実は大きな振れほど周期が長いことが判明する．すなわち，ネジを巻いた（電池をかえた）直後の柱時計は大きな振幅をとるから遅れ，止まりそうな柱時計は小さい振幅なので時

計は進むのである. 止まりそうな柱時計は遅れると考えてはいませんでしたか？

５．　二次元の運動

■　ローレンツ力　電荷qを持つ荷電粒子が電場$\boldsymbol{E}$と磁束密度$\boldsymbol{B}$から受ける力は,

$$\boldsymbol{f}(\boldsymbol{r},\boldsymbol{v},t) = q[\boldsymbol{E}(\boldsymbol{r},t) + \boldsymbol{v}\times\boldsymbol{B}(\boldsymbol{r},t)] \tag{30}$$

で与えられる．これをローレンツ力とよぶ.

■　一様電場中の運動　一様な電場$\boldsymbol{E} = (E,0,0)$があったとき, 電子は電場と反対のx軸負方向に加速される．解は易しいので省略しよう.

■　一様磁場中の運動　初速度が$(v,0,0)$の電子が，一様な磁束密度$\boldsymbol{B} = (0,0,B)$に入射した場合を考えよう．このとき磁場の方向と垂直なxy面内の電子の運動方程式だけ考えればよい．電子の電荷は$-e$だから，運動方程式を書き直すと

$$m\dot{v}_x = -ev_yB, \quad m\dot{v}_y = ev_xB \tag{31}$$

となる．サイクロトロン角速度$\omega_c = eB/m$を使い，虚数単位iを使うと

$$\dot{v}_x + i\dot{v}_y = i\omega_c(v_x + iv_y) \tag{32}$$

とまとめて書ける．したがって，解は$v_x + iv_y = Ae^{i(\omega_c t+\varphi)}$と計算される．初期条件$v_x(0) = v, v_y(0) = 0$より,

$$\begin{aligned} v_x &= v\cos\omega_c t, & v_y &= v\sin\omega_c t, \\ x &= \frac{v}{\omega_c}\sin\omega_c t + x_0, & y &= -\frac{v}{\omega_c}\cos\omega_c t + y_0 \end{aligned} \tag{33}$$

を得る．運動はサイクロトロン運動とよばれる回転運動である．ローレンツ力は速度に直交しているので仕事はせず速さは一定となる.

■　直交する電場と磁場中の運動　一様な電場$\boldsymbol{E} = (E,0,0)$とそれに直交する一様な磁束密度$\boldsymbol{B} = (0,0,B)$に入射した電子の運動を調べることにし

よう．数値計算する前に，どのようなことが起きるか想像してみよう．まず電場がないときにはサイクロトロン運動をしていた．小さな電場が加われば，当然，電場と反対方向に回転がずれて行くように考えたくなる．

数値計算では簡単のためにサイクロトロン角速度を１とし，運動方程式を

$$\dot{v}_x = -\alpha - v_y, \quad \dot{v}_y = v_x \tag{34}$$

としよう（test4.c）．

図 7-2 に$\alpha = 0.2$のときの電子の軌跡を示した．驚くべきことに，電場と垂直なy軸負方向に曲線を描きながら移動している．平均的にも電場と反対のx軸負方向に移動して行かないのである．この現象を$\boldsymbol{E} \times \boldsymbol{B}$ドリフトという．何か直観に反するような気がしませんか？

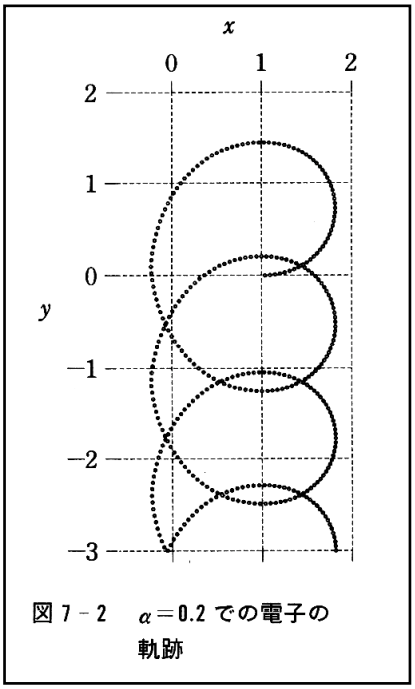

図 7－2　α＝0.2 での電子の軌跡

参考文献

１）江沢洋著：『よくわかる力学』（東京図書，1991）

２）薮下信著：『計算物理(I)』（地人書館，1982）

演習問題

表の確認　表 7-1，7-2 を確認せよ．

誤差の検討　(10)の２式の誤差がh^3であることを確かめよ．

解答：　$s_{n+1/2}$での値を中心にテイラー展開して確かめることができる．

$$x(s_{n+1}) = x(s_{n+1/2} + \frac{h}{2}) = x(s_{n+1/2}) + \frac{h}{2}\frac{\mathrm{d}x}{\mathrm{d}s}(s_{n+1/2}) + \frac{1}{2}(\frac{h}{2})^2\frac{\mathrm{d}^2x}{\mathrm{d}s^2}(s_{n+1/2}) + \cdots\cdots$$

$$x(s_n) = x(s_{n+1/2} - \frac{h}{2}) = x(s_{n+1/2}) - \frac{h}{2}\frac{\mathrm{d}x}{\mathrm{d}s}(s_{n+1/2}) + \frac{1}{2}(-\frac{h}{2})^2\frac{\mathrm{d}^2x}{\mathrm{d}s^2}(s_{n+1/2}) + \cdots\cdots$$

$$\therefore x(s_{n+1}) - x(s_n) = \frac{h}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}s}(s_{n+1/2}) + \frac{\mathrm{d}x}{\mathrm{d}s}(s_{n+1/2})\right) + O(h^3)$$

$$= \frac{h}{2}\left(v(s_{n+1/2}) + v(s_{n+1/2})\right) + O(h^3)$$

$$= \frac{h}{2}\left(v(s_n) + \frac{h}{2}\frac{\mathrm{d}v}{\mathrm{d}s}(s_n) + v(s_{n+1}) - \frac{h}{2}\frac{\mathrm{d}v}{\mathrm{d}s}(s_{n+1})\right) + O(h^3)$$

$$= \frac{h}{2}\left(v(s_n) + v(s_{n+1})\right) + (\frac{h}{2})^2(x(s_{n+1}) - x(s_n)) + O(h^3)$$

$$= \frac{h}{2}\left(v(s_n) + v(s_{n+1})\right) + (\frac{h}{2})^2(hv(s_{n+1/2})) + O(h^3)$$

$$= \frac{h}{2}\left(v(s_n) + v(s_{n+1})\right) + O(h^3)$$

ここで$O(h^3)$はh^3のオーダーであることを意味する．$h = 0.1$とすると，$h^3 = 0.001$である．vについては省略する．

単振り子の周期　単振り子の最大振幅（度）に対する周期を求めよ．

最大振幅（度）	3	5	10	20	30	40	50	60	70	80
周期（秒）				1.01					1.11	
最大振幅（度）	90	100	110	120	130	140	150	160	170	175
周期（秒）		1.24						2.02		

$\boldsymbol{E}\times\boldsymbol{B}$ドリフト　(1) (34)の$\alpha$を変化させて$\boldsymbol{E}\times\boldsymbol{B}$ドリフトの電子の軌跡を描け．(2) $\boldsymbol{E}\times\boldsymbol{B}$ドリフトを説明せよ．

解答：　(1)略．(2)この一見奇妙な数値計算の結果には理論的裏付けがないと不満足であろう．難しくなるが以下にそれを与える．$m\dot{\boldsymbol{v}} = -e(\boldsymbol{E} + \boldsymbol{v}\times\boldsymbol{B})$の方程式で与えられる運動を$\boldsymbol{u} = \boldsymbol{E}\times\boldsymbol{B}/B^2$で運動している座標系で見ると速度は$\boldsymbol{v}' = \boldsymbol{v} - \boldsymbol{u}$となる．ベクトル積の性質より$\boldsymbol{u}\times\boldsymbol{B} = (\boldsymbol{E}\times\boldsymbol{B})\times\boldsymbol{B}/B^2 = -\boldsymbol{E}$だから，

$$m\dot{\boldsymbol{v}}' = m\dot{\boldsymbol{v}} = -e(\boldsymbol{E} + \boldsymbol{v}'\times\boldsymbol{B} + \boldsymbol{u}\times\boldsymbol{B}) = -e\boldsymbol{v}'\times\boldsymbol{B}$$

と計算される．最左辺と最右辺を見れば，運動座標系では電場は消去され，磁場だけあるときの回転運動とみなせる．元の座標系では，回転運動の中心が，速度 $\boldsymbol{u} = (E,0,0)\times(0,0,B)/B^2 = (0,-E/B,0)$ で移動してゆく．(34) においては，$v_y' = v_y + \alpha$ とおくことに相当する．

x 軸負方向に引っ張る電場の効果は，電子の描く円弧のうち x 軸負側の半円では円弧の半径を大きくすることに寄与し，正側では半径を小さくすることに寄与している．

振動の計算　減衰振動，強制振動のプログラムを作り，試してみよ．

test1.c；式（６）

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
          double    h = 0.1;
          double    x,v,xp,vp;
          double    xe,ve,a;
          double    time;
          double    Pi = 3.1415926;
          int       i;

          x = 1.0; v = 1.0;
          a = sqrt(2.0);

          printf("time¥tx¥txe¥terr¥tv¥tve¥terr¥tenergy¥tEnergy¥n");

          for ( i = 0; i <= 5; i++ )
          {
                    time = i * h;
                    xe = a * sin(time+Pi/4.0);
                    ve = a * cos(time+Pi/4.0);
                    printf("%1.3f¥t",time);
                    printf("%1.3f¥t%1.3f¥t%1.3f¥t",x,xe,(x-xe)/xe*100);
                    printf("%1.3f¥t%1.3f¥t%1.3f¥t",v,ve,(v-ve)/ve*100);
                    printf("%1.3f¥t",x * x + v * v);
                    printf("%1.3f¥n",xe * xe + ve * ve);

                    xp = x; vp = v;
                    x = xp + vp * h;
                    v = vp - xp * h;
          }
          exit(0);
}
```

test2.c ; 式 （１２）

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int main()
{
          double    h = 0.1;
          double    x,v,vp,vm;
          double    xe,ve,a;
          double    time;
          double    Pi = 3.1415926;
          int       i;

          a = sqrt(2.0);
          x = 1.0; v = 1.0;
          vp = v + x * 0.1/2.0;
          v = v - x * 0.1/2.0;

          printf("time¥tx¥txe¥tv¥tve¥tenergy¥tEnergy¥n");
          for ( i = 0; i <= 100; i++ )
          {
                    time = i * h;
                    if ( i%10 == 0 )
                    {
                              vm = (v + vp ) / 2.0;
                              xe = a * sin(time+Pi/4.0);
                              ve = a * cos(time+Pi/4.0);
                              printf("%1.3f¥t",time);
                              printf("%1.3f¥t%1.3f¥t",x,xe);
                              printf("%1.3f¥t%1.3f¥t",vm,ve);
                              printf("%1.3f¥t",x * x + vm * vm);
                              printf("%1.3f¥n",xe * xe + ve * ve);
                    }
                    vp = v;
                    x = x + v * h;
                    v = v - x * h;
          }
          exit(0);
}
```

test3.c ; 単振り子の周期

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
double x0,v0;
extern double Func1(double x, double v, double t)
{
          return v;
}
extern double Func2(double x, double v, double t)
{
          return -sin(x);
}
extern void Runge(double x, double v, double t, double h)
{
          double k1,k2,k3,k4;
          double m1,m2,m3,m4;

/*  4th Order */
          k1 = h * Func1(x,v,t);
          m1 = h * Func2(x,v,t);
          k2 = h * Func1(x+k1/2,v+m1/2,t+h/2);
          m2 = h * Func2(x+k1/2,v+m1/2,t+h/2);
          k3 = h * Func1(x+k2/2,v+m2/2,t+h/2);
          m3 = h * Func2(x+k2/2,v+m2/2,t+h/2);
          k4 = h * Func1(x+k3,v+m3,t+h);
          m4 = h * Func2(x+k3,v+m3,t+h);

          x0 += (k1+k4+2*(k2+k3))/6.0;
          v0 += (m1+m4+2*(m2+m3))/6.0;

/*  Secon Order */
/*
          k2 = h * Func1(x+k1,v+m1,t+h);
          m2 = h * Func2(x+k1,v+m1,t+h);
          x0 += (k1+k2)/2.0;
          v0 += (m1+m2)/2.0;
*/
}
int main()
{
          double    h = 0.01;
          double    phi;
          double    time,kizami;
          double    Pi = 3.1415926;
          double    x1;
          int       i,i1,i2;
          int       k = 0;
          FILE      *fp;

          printf("¥n¥tphi =¥t");
          scanf("%lf",&phi);
          printf("¥n%f¥n",phi);
          x0 = phi/180.0*Pi;
          v0 = 0.0;
          kizami = h * sqrt(0.25/9.8);

          fp = fopen("furiko.dat","w");
          for ( i = 0; i <= 2000; i++ )
          {
                    time = i * h;

                    if ( x0 * x1 < 0 )
```

```
                    {
                              if ( k == 0 ) { i1 = i; k = 1; }
                              else
                              {
                                        i2 = i;
                                        printf("¥nphi = %3.1f¥tT = %f  +-  %f¥n¥n",
                                                  phi,2*(i2-i1)*kizami,2*kizami);
                                        exit(0);
                              }
                    }
                    printf("%f¥t%f¥t%f¥n",i * kizami,x0,v0);
                    fprintf(fp,"%f¥t%f¥t%f¥n",i * kizami,x0,v0);
                    x1 = x0;
                    Runge(x0,v0,time,h);
          }
          fclose(fp);
          exit(0);
}
```

test4.c ; 電子の運動

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
double x0[2],v0[2];
double f[2];
double ef;
extern void Func1(double x[2], double v[2], double t)
{
          int i;
          for ( i = 0; i < 2; i++ ) f[i] = v[i];
}
extern void Func2(double x[2], double v[2], double t)
{
          f[0] = -ef-v[1];
          f[1] = v[0];
}
extern void Runge(double x[2], double v[2], double t, double h)
{
          int i;
          double k1[2],k2[2],k3[2],k4[2];
          double m1[2],m2[2],m3[2],m4[2];
          double xx[2],vv[2];

/*  4th Order */
          Func1(x,v,t);
          for ( i = 0; i < 2; i++ ) k1[i] = h * f[i];
          Func2(x,v,t);
          for ( i = 0; i < 2; i++ ) m1[i] = h * f[i];
          for ( i = 0; i < 2; i++ )
          {
                    xx[i] = x[i] + k1[i]/2;
                    vv[i] = v[i] + m1[i]/2;
          }
          Func1(xx,vv,t+h/2);
          for ( i = 0; i < 2; i++ ) k2[i] = h * f[i];
          Func2(xx,vv,t+h/2);
          for ( i = 0; i < 2; i++ ) m2[i] = h * f[i];
          for ( i = 0; i < 2; i++ )
          {
                    xx[i] = x[i] + k2[i]/2;
                    vv[i] = v[i] + m2[i]/2;
          }
          Func1(xx,vv,t+h/2);
          for ( i = 0; i < 2; i++ ) k3[i] = h * f[i];
          Func2(xx,vv,t+h/2);
          for ( i = 0; i < 2; i++ ) m3[i] = h * f[i];
          for ( i = 0; i < 2; i++ )
          {
                    xx[i] = x[i] + k3[i];
                    vv[i] = v[i] + m3[i];
          }
          Func1(xx,vv,t+h);
          for ( i = 0; i < 2; i++ ) k4[i] = h * f[i];
          Func2(xx,vv,t+h);
          for ( i = 0; i < 2; i++ ) m4[i] = h * f[i];

          for ( i = 0; i < 2; i++ )
          {
                    x0[i] += (k1[i]+k4[i]+2*(k2[i]+k3[i]))/6.0;
                    v0[i] += (m1[i]+m4[i]+2*(m2[i]+m3[i]))/6.0;
          }
}
int main()
```

```
{
        double    h = 0.05;
        double    time;
        int       i;
        FILE      *fp;

        printf("electric field = ?¥t");
        scanf("%lf",&ef);

        x0[0] = 1.0; x0[1] = .0;
        v0[0] = 1.0; v0[1] = .0;
        fp = fopen("ebdrift.dat","w");

        for ( i = 0; i <= 500; i++ )
        {
                time = i * h;
                printf("%f¥t%f¥t%f¥t%f¥t%f¥n",time,x0[0],x0[1],v0[0],v0[1]);
                fprintf(fp,"%f¥t%f¥t%f¥t%f¥t%f¥n",time,x0[0],x0[1],v0[0],v0[1]);
                Runge(x0,v0,time,h);
        }
        fclose(fp);
        exit(0);
}
```