最終更新日: 2006 年 12 月 1 日

「C 言語による数値計算入門」(初版) 正誤表

	誤	正
p.1	β 進 t 桁の**浮動小数点数** $\bar{x}$ とは, $\bar{x} = \pm\left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_t}{\beta^t}\right) \times \beta^e$ $= \pm(0.d_1 d_2 \ldots d_t)_\beta \times \beta^e,$ $d_i \in \{0, 1, \ldots, \beta - 1\}, \quad e_{\min} \le e \le e_{\max}$ と表現される数のことです. ここで, $\pm$ を**符号**, e を**指数**, $(d_1 d_2 \ldots d_t)_\beta$ を**仮数**といいます. 浮動小数点数は指数部と仮数部の対で表されるため, 数によって小数点の位置が動きます.	β 進 t 桁の**浮動小数点数** $\bar{x}$ とは, $\bar{x} = \pm\left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_{t-1}}{\beta^{t-1}}\right) \times \beta^e$ $= \pm(d_0.d_1 d_2 \ldots d_{t-1})_\beta \times \beta^e,$ $d_i \in \{0, 1, \ldots, \beta - 1\}, \quad e_{\min} \le e \le e_{\max}$ と表現される数のことです. ここで, $\pm$ を**符号**, e を**指数**, $(d_0.d_1 d_2 \ldots d_{t-1})_\beta$ を**仮数**といいます. 浮動小数点数は指数部 β^e と仮数部 $(d_0.d_1 d_2 \ldots d_{t-1})_\beta$ の対で表されるため, 数によって小数点の位置が動きます.
p.2, (1.2)	$\bar{x} = \left(\frac{0}{10} + \frac{1}{10^2} + \frac{1}{10^3} + \frac{1}{10^4}\right) \times 10^{-1} = 0.00\underline{111}$ と表現できます. ここで, $d_1 \neq 0$ とすると, 浮動小数点数 $\tilde{x}$ は $\tilde{x} = \left(\frac{1}{10} + \frac{1}{10^2} + \frac{1}{10^3} + \frac{1}{10^4}\right) \times 10^{-2} = 0.00\underline{1111}$ となり, $\tilde{x}$ の方が有効桁数が増えていることが分かります.	$\bar{x} = \left(0 + \frac{0}{10} + \frac{1}{10^2} + \frac{1}{10^3}\right) \times 10^{-1} = 0.00\underline{11}$ と表現できます. ここで, $d_1 \neq 0$ とすると, 浮動小数点数 $\tilde{x}$ は $\tilde{x} = \left(1 + \frac{1}{10} + \frac{1}{10^2} + \frac{1}{10^3}\right) \times 10^{-3} = 0.00\underline{1111}$ となり, $\tilde{x}$ の方が有効桁数が増えていることが分かります.
p.2, 中程	$x + y = (0.43251 + 0.0000023446) \times 10^3 = 0.43251 \times 10^3 = x$	$x + y = (4.3251 + 0.000023446) \times 10^2 = 4.3251 \times 10^2 = x$
p.2, 下	また, $x = \sqrt{10001} \approx 100.005 = 0.100005 \times 10^3$, $y = \sqrt{10000} = 100 = 0.100000 \times 10^3$ のとき, $x - y$ を 10 進有効桁数 6 桁の四捨五入で計算すると, $x - y = (0.100005 - 0.100000) \times 10^3 = (0.000005) \times 10^3 = 0.\underline{5} \times 10^{-2}$	また, $x = \sqrt{10001} \approx 100.005 = 1.00005 \times 10^2$, $y = \sqrt{10000} = 100 = 1.00000 \times 10^2$ のとき, $x - y$ を 10 進有効桁数 6 桁の四捨五入で計算すると, $x - y = (1.00005 - 1.00000) \times 10^2 = (0.00005) \times 10^2 = \underline{5.0} \times 10^{-3}$

	誤	正
p.3, 上	$\sqrt{10001}-\sqrt{10000} = \frac{1}{\sqrt{10001}+\sqrt{10000}} = \frac{1}{(0.100005+0.100000)\times 10^3}$ $= \frac{(0.100000)\times 10}{(0.200005)\times 10^3} = \frac{0.100000}{0.200005}\times 10^{1-3}$ $= 0.499999\times 10^{-2}$	$\sqrt{10001}-\sqrt{10000} = \frac{1}{\sqrt{10001}+\sqrt{10000}} = \frac{1}{(0.100005+0.100000)\times 10^3}$ $= \frac{(1.00000)\times 10^{-1}}{(2.00005)\times 10^2} = \frac{1.00000}{2.00005}\times 10^{-1-2}$ $= \underline{4.99988}\times 10^{-3}$
p.3, 下	$(x+y)+z = (0.537+0.612)+0.128 = 1.14+0.128 = 1.26,$ $x+(y+z) = 0.537+(0.612+0.128) = 0.537+0.74 = 1.27$	$(x+y)+z = (5.37+6.12)\times 10^{-1}+1.28\times 10^{-1} = (1.14+0.128)\times 10^0 = 1.26,$ $x+(y+z) = 5.37\times 10^{-1}+(6.12+1.28)\times 10^{-1} = (5.37+7.4)\times 10^{-1} = 1.27$
p.5, 表 1.1	$e_{\max}$ | $+127$ | $\geq +1023$ | $+1023$ | $> +16383$	$e_{\max}$ | $+127$ | $\geq +1023$ | $+1023$ | $\geq +16383$
p.5, 表 1.1 の下	浮動小数点数フォーマット (1.1) において，$d_1 = 1$ とすることに対応します.	浮動小数点数フォーマット (1.1) において，$d_0 = 1$ とすることに対応します.
p.7, プログラム 1.3	`#include <stdio.h>`	`#include <stdio.h>` `#include <string.h>`
p.10, 無限大	$x_{\max} = (0.111\ldots1)_2\times 2^{1023} = (1-2^{-53})\times 2^{1023} \approx 2^{1023}$ $x_{\min} = (0.100\ldots0)_2\times 2^{-1022} = 2^{-1022-1} = 2^{-1023}$	$x_{\max} = (1.11\ldots1)_2\times 2^{1023} = (2-2^{-52})\times 2^{1023} \approx 2^{1024}$ $x_{\min} = (1.00\ldots0)_2\times 2^{-1022} = 1.0\times 2^{-1022} = 2^{-1022}$
p.12, 最近点への丸め, 脚注を挿入，これ以降の脚注番号をずらす	もしも，このような点が 2 点ある場合は，仮数部の最終ビットが偶数である浮動小数点数に丸めます.	もしも，このような点が 2 点ある場合[15] は，仮数部の最終ビットが偶数である浮動小数点数に丸めます. †[15] 具体的には，x が 2 つの浮動小数点数 x_0 と x_1 の中点になっている場合です.

	誤	正
p.12, $1+u$ の計算	$$\begin{aligned} 1+u &= 1+2^{-53} \\ &= \left(\frac{1}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{53}}\right)\times 2+\left(\frac{1}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{53}}\right)\times 2^{-52} \\ &= \left(\frac{1}{2}+\frac{0}{2^2}+\cdots\frac{0}{2^{53}}+\frac{1}{2^{54}}\right)\times 2 \end{aligned}$$ ここで，$t=53$ であり，デフォルトでは丸めモードが最近点への丸めになっているので，$\frac{1}{2^{54}}$ は偶数の丸めにより切り捨てられます.	$$\begin{aligned} 1+u &= 1+2^{-53} \\ &= \left(1+\frac{0}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{52}}\right)\times 2^0+\left(1+\frac{0}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{52}}\right)\times 2^{-53} \\ &= \left(1+\frac{0}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{52}}+\frac{1}{2^{53}}\right)\times 2^0 \end{aligned}$$ ここで，$t=53$ であり，デフォルトでは丸めモードが最近点への丸めになっているので，$\frac{1}{2^{53}}$ は偶数の丸めにより切り捨てられます.
p.12, (1.3)	$$1+u=\left(\frac{1}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{53}}\right)\times 2=1 \qquad (1.3)$$ となります.	$$1+u=\left(1+\frac{0}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{52}}\right)\times 2^0=1 \qquad (1.3)$$ となります．なお，$1+2^{-53}=\frac{1}{2}(2+2^{-52})=\frac{1}{2}(1+1+2^{-52})$ なので $1+2^{-53}$ は 1 と $1+2^{-52}$ の中点であることが分かります．この 2 つの数のうち，仮数部の最終ビットが 0 であるのは 1 なので，このことからも $1+u=1$ となることが分かります.
p.13, $1-u$ の計算	$$\begin{aligned} 1-u &= 1-2^{-53} \\ &= \left(\frac{1}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{53}}\right)\times 2-\left(\frac{1}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{53}}\right)\times 2^{-52} \\ &= \left(\frac{1}{2}+\frac{0}{2^2}+\cdots\frac{0}{2^{53}}-\frac{1}{2^{54}}\right)\times 2 \\ &= \left(\frac{0}{2}+\frac{1}{2^2}+\cdots\frac{1}{2^{53}}\right)\times 2 \\ &= \left(\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^{52}}+\frac{0}{2^{53}}\right)\times 2^0 \\ &= \left(\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^{52}}\right)\times 2^0 \end{aligned}$$ となります.	$$\begin{aligned} 1-u &= 1-2^{-53} \\ &= \left(1+\frac{0}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{52}}\right)\times 2^0-\left(1+\frac{0}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{52}}\right)\times 2^{-53} \\ &= \left(1+\frac{0}{2}+\frac{0}{2^2}+\cdots\frac{0}{2^{52}}-\frac{1}{2^{53}}\right)\times 2^0=\left(0+\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^{52}}\right)\times 2^0 \\ &= \left(1+\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^{51}}+\frac{0}{2^{52}}\right)\times 2^{-1} \end{aligned}$$ となります．この計算では $1+u$ の場合と異なり，丸め操作を行っていないことに注意して下さい．このように丸め操作を行っていない場合は，$1+u$ のときと同様にして「$1-2^{-53}=\frac{1}{2}(2-2^{-52})=\frac{1}{2}(1+1-2^{-52})$ であり，$1-2^{-53}$ は 1 と $1-2^{-52}$ の中点なので仮数部の最終ビットが 0 である 1 に丸める」と考えてはいけません．丸め操作をしていないのに丸めを考えるのはおかしいですよね？ また，$1-2^{-52}$ の仮数部の最終ビットも 0 であることにも注意してください.

	誤	正
p.13, マシンイプシロンの 6 行目	マシンイプシロンは，$1+\varepsilon_M > 1$ を満たす最小の正数です.	マシンイプシロンは，$1+\varepsilon_M > 1$ を満たす 2^n (n は整数) の形をした最小の正数です.
p.39, (3) の直前	前進消去ぜんしんしょうきょと呼びます.	**前進消去**と呼びます.
p.65, プログラム 3.4 の main 関数内	`int i, j;`	`int i;`
p.88, (4.31) の下	ヤコビ行列の逆行列を選んだものなっていること	ヤコビ行列の逆行列を選んだものになっていること

p.98, jacobi_lin 関数の後半にある if 文．free_dvector 関数を if 文の前に移動する．また，exit(1) を追加する.	`if ( k == KMAX )`
	`{`
	`  printf("答えが見つかりませんでした\n");`
	`}`
	`else`
	`{`
	`  printf("反復回数は%d 回です\n", k); /* 反復回数を画面に表示 */`
	`  return x;`
	`}`
	`free_dvector( xn, 1 ); /* 領域の解放 */`
		`free_dvector( xn, 1 ); /* 領域の解放 */`
		`if ( k == KMAX )`
		`{`
		`  printf("答えが見つかりませんでした\n");`
		`  exit(1);`
		`}`
		`else`
		`{`
		`  printf("反復回数は%d 回です\n", k); /* 反復回数を画面に表示 */`
		`  return x;`
		`}`

p.103, gauss_seidel 関数の後半にある if 文．free_dvector 関数を if 文の前に移動する．また，exit(1) を追加する.	`if ( k == KMAX )`
	`{`
	`  printf("答えが見つかりませんでした\n");`
	`}`
	`else`
	`{`
	`  printf("反復回数は%d 回です\n", k); /* 反復回数を画面に表示 */`
	`  return x;`
	`}`
	`free_dvector( xo, 1 ); /* 領域の解放 */`
		`free_dvector( xo, 1 ); /* 領域の解放 */`
		`if ( k == KMAX )`
		`{`
		`  printf("答えが見つかりませんでした\n");`
		`  exit(1);`
		`}`
		`else`
		`{`
		`  printf("反復回数は%d 回です\n", k); /* 反復回数を画面に表示 */`
		`  return x;`
		`}`

<table>
<tr><th></th><th>誤</th><th>正</th></tr>
<tr><td>p.107, sor 関数の後半にある if 文. free_dvector 関数を if 文の前に移動する. また, exit(1) を追加する.</td><td>

```
if ( k == KMAX )
{
  printf("答えが見つかりませんでした\n");
}
else
{
  printf("反復回数は%d 回です\n", k); /* 反復回数を画面に表示 */
  return x;
}
free_dvector( xo, 1 ); /* 領域の解放 */
```

</td><td>

```
free_dvector( xo, 1 ); /* 領域の解放 */

if ( k == KMAX )
{
  printf("答えが見つかりませんでした\n");

  exit(1);

}
else
{
  printf("反復回数は%d 回です\n", k); /* 反復回数を画面に表示 */
  return x;
}
```

</td></tr>
<tr><td>p.113, cg 関数の後半にある if 文. free_dvector 関数を if 文の前に移動する. また, exit(1) を追加する.</td><td>

```
OUTPUT:;
if ( k == KMAX )
{
  printf("答えが見つかりませんでした\n");
}
else
{
  printf("反復回数は%d 回です\n", k); /* 反復回数を画面に表示 */
  return x;
}
/* 領域の解放 */
free_dvector( r, 1 ); free_dvector( p, 1 ); free_dvector( tmp, 1 );
```

</td><td>

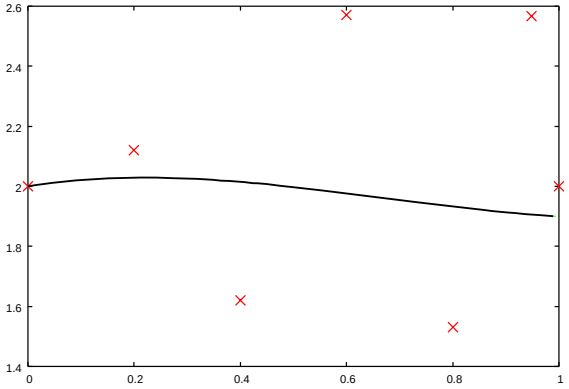
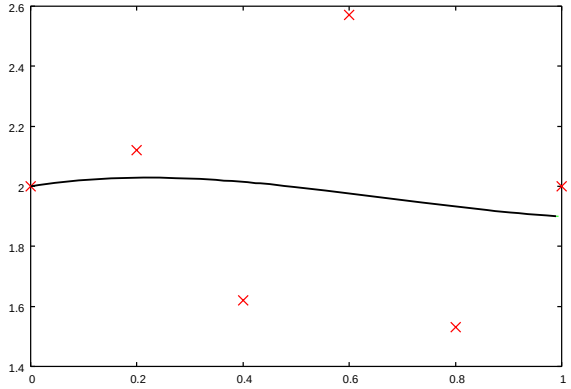
```
  OUTPUT:;
/* 領域の解放 */
free_dvector( r, 1 ); free_dvector( p, 1 ); free_dvector( tmp, 1 );

if ( k == KMAX )
{
  printf("答えが見つかりませんでした\n");

  exit(1);

}
else
{
  printf("反復回数は%d 回です\n", k); /* 反復回数を画面に表示 */
  return x;
}
```

</td></tr>
</table>

	誤	正

p.120, プログラム 6.1 の main 関数内から `int i;` を削除.	`FILE *fin, *fout;`
	`double *x, *y;`
	`int i;`
		`FILE *fin, *fout;`
		`double *x, *y;`

p.121 の最後から 4 行目. 間違いではないが, 表示を見やすくするために改行を挿入.	`/* 結果の出力 */`
	`fprintf( fout, "最小 2 乗近似式は y=\n");`
	`for( i = N+1 ; i >= 1 ; i--)`
	`{`
	`  fprintf(fout, "+ %5.2f x^%d ", a[i],i-1);`
	`}`
		`/* 結果の出力 */`
		`fprintf( fout, "最小 2 乗近似式は y=\n");`
		`for( i = N+1 ; i >= 1 ; i--)`
		`{`
		`  fprintf(fout, "+ %5.2f x^%d ", a[i],i-1);`
		`}`
		`fprintf(fout, "\n");`

p.123, 図 6.2, 右から 2 つ目の×を削除		

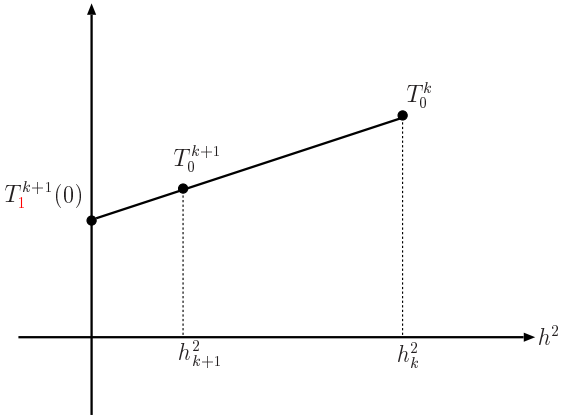
	誤	正

p.126, プログラム 6.2 の main 関数内から `int i;` を削除.	`FILE *fin, *fout;`
	`double *x, *y, xi;  /* xi は補間点 */`
	`int i, n;`
		`FILE *fin, *fout;`
		`double *x, *y, xi;  /* xi は補間点 */`
		`int n;`

p.128, 式 (6.16) の下	を満たす $G(x)$ を求めることを考えます．そのために，$x \in [a,b]$ を固定して，$x \in [a,b]$ の関数	を満たす $G(x)$ を求めることを考えます．そのために，$x \in [a,b]$ を固定して，$z \in [a,b]$ の関数
p.128 の最終行	一方，$\omega_{n+1}(x) = x^{n+1} + a_n x^n + a_{n-1}x^{n-1} + \cdots a_1 x + a_0$ と書けることに注意すれば，$\omega_n(x)^{(n+1)}(\xi) = (n+1)!$ となることが分かります．よって，	一方，$\omega_{n+1}(x) = x^{n+1} + a_n x^n + a_{n-1}x^{n-1} + \cdots a_1 x + a_0$ と書けることに注意すれば，$\omega_{n+1}(x)^{(n+1)}(\xi) = (n+1)!$ となることが分かります．よって，
p.129 の上	$$f(z) - P_n(z) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)\omega_{n+1}(z)$$ が成り立ちます.	$$f(x) - P_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)\omega_{n+1}(x)$$ が成り立ちます.
p.131, 式 (6.22)	$P_n(x) = [f_0] + (x - x_0)f[x_0, x_1]$	$P_n(x) = f[x_0] + (x - x_0)f[x_0, x_1]$

p.132, プログラム 6.3 の main 関数内から `int i;` を削除.	`FILE *fin, *fout;`
	`double *x, *y, xi;  /* xi は補間点 */`
	`int i, n;`
		`FILE *fin, *fout;`
		`double *x, *y, xi;  /* xi は補間点 */`
		`int n;`

p.142, 式 (7.10)	$$\begin{aligned}\int_a^b f(x)dx &= \int_{x_0}^{x_2} f(x)dx = \int_{-h}^{h} f(x_1+z)dz \\ &= \int_{-h}^{h} \left(f(x_1) + zf'(x_1) + \frac{z^2}{2!}f''(x_1) + \frac{z^3}{3!}f'''(x_1) + \frac{z^4}{4!}f^{(4)}(x_1) + \cdots \right) dz \\ &= 2hf(x_1) + \frac{h^3}{3}f''(x_1) + \frac{h^5}{60}f^{(4)}(x_1) + \frac{h^7}{360}f^{(6)}(x_1) + \cdots\end{aligned}$$	$$\begin{aligned}\int_a^b f(x)dx &= \int_{x_0}^{x_2} f(x)dx = \int_{-h}^{h} f(x_1+z)dz \\ &= \int_{-h}^{h} \left(f(x_1) + zf'(x_1) + \frac{z^2}{2!}f''(x_1) + \frac{z^3}{3!}f'''(x_1) + \frac{z^4}{4!}f^{(4)}(x_1) + \cdots \right) dz \\ &= 2hf(x_1) + \frac{h^3}{3}f''(x_1) + \frac{h^5}{60}f^{(4)}(x_1) + \frac{h^7}{2520}f^{(6)}(x_1) + \cdots\end{aligned}$$

	誤	正
p.143, 式 (7.11)	$$f''(x_1)=\frac{f(x_0)-2f(x_1)+f(x_2)}{h^2}-\frac{h^4}{12}f^{(4)}(x_1)-\frac{h^6}{360}f^{(6)}(x_1)+\cdots \qquad (7.11)$$ となります．(7.11) を (7.10) に代入すると $$\int_a^b f(x)dx=\frac{h}{3}(f(x_0)+4f(x_1)+f(x_2))-\frac{h^5}{90}f^{(4)}(x_1)+\frac{h^7}{540}f^{(6)}(x_1)+\cdots$$	$$f''(x_1)=\frac{f(x_0)-2f(x_1)+f(x_2)}{h^2}-\frac{h^2}{12}f^{(4)}(x_1)-\frac{h^4}{360}f^{(6)}(x_1)+\cdots \qquad (7.11)$$ となります．(7.11) を (7.10) に代入すると $$\int_a^b f(x)dx=\frac{h}{3}(f(x_0)+4f(x_1)+f(x_2))-\frac{h^5}{90}f^{(4)}(x_1)-\frac{h^7}{1890}f^{(6)}(x_1)+\cdots$$
p.148, 式 (7.15)	$$\begin{aligned} T_0^k-I &\approx \frac{b-a}{12}M_1h_k^2 \\ T_0^{k+1}-I &\approx \frac{b-a}{12}M_1h_{k+1}^2=\frac{b-a}{12}M_1\frac{h_k^2}{4} \end{aligned} \qquad (7.15)$$ なので，$\frac{b-a}{12}M_1$ を消去すると	$$\begin{aligned} \lvert T_0^k-I\rvert &\approx \frac{b-a}{12}M_1h_k^2 \\ \lvert T_0^{k+1}-I\rvert &\approx \frac{b-a}{12}M_1h_{k+1}^2=\frac{b-a}{12}M_1\frac{h_k^2}{4} \end{aligned} \qquad (7.15)$$ なので，$T_0^k-I\geq 0$, $T_0^{k+1}-I\geq 0$ として $\frac{b-a}{12}M_1$ を消去すると
p.148 の図		

	誤	正
p.152, 第 3 行以降	まず，$x_0 < x_1$, $f_0 = f(x_0)$, $f_1 = f(x_1)$, $h = x_1 - x_0$, $I = \int_{x_0}^{x_1} f(x)dx$ とし，台形公式の誤差評価を再考してみます． テイラー展開 $$f_1 = f_0 + hf_0' + \frac{h^2}{2!}f_0'' + \frac{h^3}{3!}f_0''' + \frac{h^4}{4!}f_0^{(4)} + \cdots$$ より $$f_0' = \frac{f_1 - f_0}{h} - \frac{h}{2}f_0'' - \frac{h^2}{6}f_0''' - \frac{h^4}{24}f_0^{(4)} - \cdots \quad (7.18)$$ なので， $$\begin{aligned}\int_{x_0}^{x_1} f(x)dx &= \int_0^h f(x_0+z)dz \\ &= \int_0^h \left(f_0 + zf_0' + \frac{z^2}{2!}f_0'' + \frac{z^3}{3!}f_0''' + \frac{z^4}{4!}f_0^{(4)} + \cdots\right)dz \\ &= hf_0 + \frac{h^2}{2}f_0' + \frac{h^3}{6}f_0'' + \frac{h^4}{24}f_0''' + \frac{h^5}{120}f_0^{(4)} + \cdots \quad (7.19)\end{aligned}$$ となります．ここで，(7.18) と (7.19) より， $$\int_{x_0}^{x_1} f(x)dx = h\frac{f_0+f_1}{2} - \frac{h^3}{12}f_0'' - \frac{h^5}{120}f_0^{(4)} - \cdots$$ を得ます．これより，台形公式による積分を $T_0 := h\frac{f_0+f_1}{2}$ とすると，(7.7) は	まず，$x_0 < x_1$, $h = x_1 - x_0$, $I = \int_{x_0}^{x_1} f(x)dx$ とし，台形公式の誤差評価を再考するために，変数変換 $z = x - \frac{x_0+x_1}{2}$ を行います．このとき，$x_0 \le x \le x_1$ ならば，$-\frac{h}{2} \le z \le \frac{h}{2}$ となります． マクローリン展開より $$\begin{aligned}f(z) &= f(0) + zf'(0) + \frac{z^2}{2!}f''(0) + \frac{z^3}{3!}f'''(0) + \frac{z^4}{4!}f^{(4)}(0) + \cdots \\ f\left(\frac{h}{2}\right) &= f(0) + \frac{h}{2}f'(0) + \frac{h^2}{8}f''(0) + \frac{h^3}{48}f'''(0) + \frac{h^4}{384}f^{(4)}(0) + \cdots \\ f\left(-\frac{h}{2}\right) &= f(0) - \frac{h}{2}f'(0) + \frac{h^2}{8}f''(0) - \frac{h^3}{48}f'''(0) + \frac{h^4}{384}f^{(4)}(0) + \cdots\end{aligned}$$ であり，台形公式による数値積分値 T_0 は $$\begin{aligned}T_0 &:= \frac{h}{2}\left(f\left(\frac{h}{2}\right) + f\left(-\frac{h}{2}\right)\right) \\ &= \frac{h}{2}\left(2f(0) + \frac{h^2}{4}f''(0) + \frac{h^4}{192}f^{(4)}(0) + \cdots\right) \\ &= hf(0) + \frac{h^3}{8}f''(0) + \frac{h^5}{384}f^{(4)}(0) + \cdots \quad (7.18)\end{aligned}$$ と表すことができます．一方， $$\begin{aligned}\int_{x_0}^{x_1} f(x)dx &= \int_{-\frac{h}{2}}^{\frac{h}{2}} f(z)dz \\ &= \int_{-\frac{h}{2}}^{\frac{h}{2}} \left(f(0) + zf'(0) + \frac{z^2}{2!}f''(0) + \frac{z^3}{3!}f'''(0) + \frac{z^4}{4!}f^{(4)}(0) + \cdots\right)dz \\ &= hf(0) + \frac{h^3}{24}f''(0) + \frac{h^5}{1920}f^{(4)}(0) + \cdots \quad (7.19)\end{aligned}$$ となります．ここで，(7.18) と (7.19) より $$\begin{aligned}\int_{x_0}^{x_1} f(x)dx - T_0 &= \left(hf(0) + \frac{h^3}{24}f''(0) + \frac{h^5}{1920}f^{(4)}(0) + \cdots\right) \\ &\quad - \left(hf(0) + \frac{h^3}{8}f''(0) + \frac{h^5}{384}f^{(4)}(0) + \cdots\right) \\ &= -\frac{h^3}{12}f''(0) - \frac{h^5}{480}f^{(4)}(0) - \cdots\end{aligned}$$ を得ます．これより，(7.7) は

<table>
<tr><th></th><th>誤</th><th>正</th></tr>
<tr><td>p.152 の (7.21)</td><td>$$\begin{aligned} T_0^k - I &\approx \frac{b-a}{12} M_1 h_k^2 + O(h_k^4) \\ T_0^{k+1} - I &\approx \frac{b-a}{12} M_1 h_{k+1}^2 + O(h_{k+1}^4) \end{aligned}$$</td><td>$$\begin{aligned} |T_0^k - I| &\approx \frac{b-a}{12} M_1 h_k^2 + O(h_k^4) \\ |T_0^{k+1} - I| &\approx \frac{b-a}{12} M_1 h_{k+1}^2 + O(h_{k+1}^4) \end{aligned}$$</td></tr>
<tr><td>p.155 の (8.9) の右辺第 1 項</td><td>$$\frac{y_{i+1} - (\alpha_0 y_1 + \cdots + \alpha_{k-1} y_{i-k+1})}{h}$$</td><td>$$\frac{y_{i+1} - (\alpha_0 y_i + \cdots + \alpha_{k-1} y_{i-k+1})}{h}$$</td></tr>
<tr><td>p.158</td><td>$$\begin{aligned} y_{i+1} &= y_i + h y'(x_i) + \frac{h^2}{2} y''(x_i) + O(h^3) \\ &= y_i + h f(x_i, y_i) + \frac{h^2}{2} \left(\frac{f(x_i + h, y_i + f(x_i, y_i)) - f(x_i, y_i) - O(h^2)}{h} \right) + O(h^3) \\ &= y_n + \frac{h}{2} \left(f(x_i, y_i) + f(x_{i+1}, y_i + h f(x_i, y_i)) \right) + O(h^3) \end{aligned}$$</td><td>$$\begin{aligned} y_{i+1} &= y_i + h y'(x_i) + \frac{h^2}{2} y''(x_i) + O(h^3) \\ &= y_i + h f(x_i, y_i) + \frac{h^2}{2} \left(\frac{f(x_i + h, y_i + h f(x_i, y_i)) - f(x_i, y_i) - O(h^2)}{h} \right) + O(h^3) \\ &= y_i + \frac{h}{2} \left(f(x_i, y_i) + f(x_{i+1}, y_i + h f(x_i, y_i)) \right) + O(h^3) \end{aligned}$$</td></tr>
<tr><td>p.160, ルンゲ・クッタ法のアルゴリズムの最後, フォントを変更</td><td>end for</td><td>end for</td></tr>
<tr><td>p.161 の main 関数の前</td><td><pre>/* ルンゲ・クッタ法 */
double *rk4(double y0, double a, double b, int n, double (*f)());</pre></td><td><pre>/* ルンゲ・クッタ法 */
double *rk4(double y0, double *y, double a, double b, int n, double (*f)());</pre></td></tr>
<tr><td>p.161 の main 関数内</td><td><pre>y = dvector(0, n);   /* 領域の確保 */
y = rk4(y0, a, b, n, func); /* ルンゲ・クッタ法 */

/* 結果の表示 */
h = (b-a)/n ;   /* 刻み幅 */
for (i = 0 ; i <= n ; i++)
{
  printf("x=%f \t y=%f \n", a+i*h, y[i]);
}

return 0;</pre></td><td><pre>y = dvector(0, n);   /* 領域の確保 */
y = rk4(y0, y, a, b, n, func); /* ルンゲ・クッタ法 */

/* 結果の表示 */
h = (b-a)/n ;   /* 刻み幅 */
for (i = 0 ; i <= n ; i++)
{
  printf("x=%f \t y=%f \n", a+i*h, y[i]);
}
free_dvector(y, 0); /* 領域の解放 */
return 0;</pre></td></tr>
</table>

<table>
<tr><th></th><th>誤</th><th>正</th></tr>
<tr><td>p.161 の rk4 関数の内．領域確保・解放を行わない．</td><td>

```
/* ルンゲ・クッタ法 */
double *rk4( double y0, double a, double b, int n, double (*f)() );
{
  double k1, k2, k3, k4, h, *y, x;
  int i;

  y = dvector( 0, n );   /* y[0,1,...n] の確保 */

    ⋮

  return y;
  free_dvector( y, 0 ); /* 領域の解放 */
}
```

</td><td>

```
/* ルンゲ・クッタ法 */
double *rk4( double y0, double *y, double a, double b, int n, double (*f)() );
{
  double k1, k2, k3, k4, h, x;
  int i;

    ⋮

  return y;

}
```

</td></tr>
<tr><td>p.169</td><td>

$$f(x,y(x)) - P_{k-1}(x) = \frac{1}{k!} f^{(k)}(\xi)\omega_{k-1}(x)$$

となるので，

$$\begin{aligned} r &:= \int_{x_i}^{x_{i+1}} (f(x,y(x)) - P_{k-1}(x))dx = \frac{1}{k!} f^{(k)}(\xi) \int_{x_i}^{x_{i+1}} \omega_{k-1}(x)dx \\ &= \frac{1}{k!} f^{(k)}(\xi) \int_{x_i}^{x_{i+1}} (x-x_i)(x-x_{i-1})\cdots(x-x_{i-k+1})dx \end{aligned}$$

となります．ここで，$x - x_i = ht$ とすれば，$x_i = x_{i-1} + h$ より

</td><td>

$$f(x,y(x)) - P_{k-1}(x) = \frac{1}{k!} f^{(k)}(\xi)\omega_{k}(x)$$

となるので，

$$\begin{aligned} r &:= \int_{x_i}^{x_{i+1}} (f(x,y(x)) - P_{k-1}(x))dx = \frac{1}{k!} f^{(k)}(\xi) \int_{x_i}^{x_{i+1}} \omega_{k}(x)dx \\ &= \frac{1}{k!} f^{(k)}(\xi) \int_{x_i}^{x_{i+1}} (x-x_i)(x-x_{i-1})\cdots(x-x_{i-k+1})dx \end{aligned}$$

となります．ここで，$x - x_i = ht$ とすれば，$x_i = x_{i-1} + h$ より

</td></tr>
<tr><td>p.172 の main 関数の前</td><td>

```
/* ルンゲ・クッタ法 */
double *rk4( double y0, double a, double b, int n, double (*f)() );
/* アダムス法 */
double *adams( double y0, double a, double b, int n,
                          int N, double eps, double (*f)() );
```

</td><td>

```
/* ルンゲ・クッタ法 */
double *rk4( double y0, double *y, double a, double b, int n, double (*f)() );
/* アダムス法 */
double *adams( double y0, double *y,  double a, double b, int n, int N, double eps,
double (*f)() );
```

</td></tr>
</table>

<table>
<tr><th></th><th>誤</th><th>正</th></tr>
<tr><td>p.172 の main 関数内</td><td>

```
y = dvector( 0, n );                /* 領域の確保 */

/* アダムス法 */
y = adams( y0, a, b, n, N, eps, func );

/* 結果の表示 */
h = (b-a)/n ;              /* 刻み幅 */
for ( i = 0 ; i <= n ; i++)
{
  printf("x=%f \t y=%f \n", a+i*h, y[i] );
}

return 0;
```

</td><td>

```
y = dvector( 0, n );                /* 領域の確保 */

/* アダムス法 */
y = adams( y0,y, a, b, n, N, eps, func );

/* 結果の表示 */
h = (b-a)/n ;              /* 刻み幅 */
for ( i = 0 ; i <= n ; i++)
{
  printf("x=%f \t y=%f \n", a+i*h, y[i] );
}
free_dvector( y, 0 ); /* 領域の解放 */
return 0;
```

</td></tr>
<tr><td>p.173, adams 関数の free_dvector 関数の位置を return 文の前へ. また,「解法」を「解放」へ.</td><td>

```
double *adams( double y0, double a, double b, int n,
                          int N, double eps, double (*f)() )
{
  double yp, h, *y, *F, x;
  int i, j;

  y = dvector( 0, n );     /* y[0,1,...n] の確保 */
  F = dvector( 0, 4 );    /* F[0,1,...4] の確保 */

    ⋮

  return y;
  free_dvector( y, 0 ); free_dvector( F, 0 ); /* 領域の解法 */
```

</td><td>

```
double *adams( double y0, double *y,  double a, double b, int n,
int N, double eps, double (*f)() )
{
  double yp, h, *F, x;
  int i, j;

  F = dvector( 0, 4 );    /* F[0,1,...4] の確保 */
        ⋮
  free_dvector( F, 0 ); /* 領域の解放 */
  return y;
```

</td></tr>
</table>

	誤	正
p.173, adams 関数内	`y = rk4( y0, a, b, n, f);`	`y = rk4( y0, y, a, b, n, f);`
p.179 の最後	`/* 境界値問題を解く */` `double *bvp(double a1, double a2, double u0, double un, int n, double (*f)() );`	`/* 境界値問題を解く */` `double *bvp( double *b, double a1, double a2, double u0, double un, int n, double (*f)() );`
p.180 の main 関数の直前から, ガウス消去法の記述を削除	`/* 部分ピボット選択付きガウス消去法 */` `double *gauss( double **a, double *b, int n );`	
p.180 の main 関数内	`u = bvp( 0.0, 1.0, 0.0, 0.0, n, func );`	`u = bvp( u, 0.0, 1.0, 0.0, 0.0, n, func );`

p.181, bvp 関数の free_dmatrix 関数の位置を return 文の前へ.	`double *bvp(double a1, double a2, double u0, double un, int n, double (*f)() )`
	`{`
	`  double h, h2, **a, *b;  /* 区間は [a1, a2] */`
	`  int i, j;`
	`  h = (a2-a1)/n;  /* 刻み幅 */`
	`  h2 = h*h;`
	`  a = dmatrix( 1, n-1, 1, n-1 ); /* 係数行列 */`
	`  b = dvector( 1, n-1 );  /* 右辺ベクトル */`
	`  :`
	`  :`
	`  return b;`
	`  /* 領域の解放 */`
	`  free_dmatrix( a, 1, n-1, 1, n-1); free_dvector( b, 1 );`
		`double *bvp( double *b, double a1, double a2, double u0, double un, int n, double (*f)() )`
		`{`
		`  double h, h2, **a; /* 区間は [a1, a2] */`
		`  int i, j;`
		`  h = (a2-a1)/n;  /* 刻み幅 */`
		`  h2 = h*h;`
		`  a = dmatrix( 1, n-1, 1, n-1 ); /* 係数行列 */`
		`  :`
		`  :`
		`  /* 領域の解放 */`
		`  free_dmatrix( a, 1, n-1, 1, n-1);`
		`  return b;`

	誤	正
p.184，定理 9.1 の証明の最後.	(9.8) より，$\frac{\boldsymbol{x}^{(k)}}{c_1\lambda_1^k}$ が λ_1 の固有値 $\boldsymbol{u}_1$ に近付き	(9.8) より，$\frac{\boldsymbol{x}^{(k)}}{c_1\lambda_1^k}$ が λ_1 の固有ベクトル $\boldsymbol{u}_1$ に近付き
p.185，式 (9.11) の 2 行目	$= \dfrac{(\lambda_1^k c_1(\boldsymbol{u}_1+e(k)), \lambda_1^{k+1} c_1(\boldsymbol{u}_1+e(k+1)))}{(\lambda_1^k c_1(\boldsymbol{u}_1, e(k)), \lambda_1^k c_1(\boldsymbol{u}_1+e(k)))}$	$= \dfrac{(\lambda_1^k c_1(\boldsymbol{u}_1+e(k)), \lambda_1^{k+1} c_1(\boldsymbol{u}_1+e(k+1)))}{(\lambda_1^k c_1(\boldsymbol{u}_1+e(k)), \lambda_1^k c_1(\boldsymbol{u}_1+e(k)))}$
p.186，べき乗法のアルゴリズム	`while` $\|\boldsymbol{v}\|_2^2 - \|\lambda^{(k)}\|^2 < \varepsilon$	`while` $\|\boldsymbol{v}\|_2^2 - \|\lambda^{(k)}\|^2 \geq \varepsilon$
p.191, 補題 9.1 の証明の 3 行目	$(E_n - 2\boldsymbol{u}\boldsymbol{u}^t)\boldsymbol{x} = \boldsymbol{x} - \dfrac{2(\boldsymbol{x}-\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y})^t\boldsymbol{x}}{(\boldsymbol{x}-\boldsymbol{y})^t(\boldsymbol{x}-\boldsymbol{y})} = \boldsymbol{x} - \dfrac{2(\boldsymbol{x}-\boldsymbol{y})(\boldsymbol{x}^t\boldsymbol{x}-\boldsymbol{y}^t\boldsymbol{x})}{\boldsymbol{x}^t\boldsymbol{x}-\boldsymbol{y}^t\boldsymbol{x}-\boldsymbol{x}^t\boldsymbol{y}+\boldsymbol{y}^t\boldsymbol{y}}$ $= \boldsymbol{x} - \dfrac{2(\boldsymbol{x}^t\boldsymbol{x}-\boldsymbol{y}^t\boldsymbol{x})}{2(\boldsymbol{x}^t\boldsymbol{x}-2\boldsymbol{y}^t\boldsymbol{x})}(\boldsymbol{x}-\boldsymbol{y}) = \boldsymbol{y}$	$(E_n - 2\boldsymbol{u}\boldsymbol{u}^t)\boldsymbol{x} = \boldsymbol{x} - \dfrac{2(\boldsymbol{x}-\boldsymbol{y})(\boldsymbol{x}-\boldsymbol{y})^t\boldsymbol{x}}{(\boldsymbol{x}-\boldsymbol{y})^t(\boldsymbol{x}-\boldsymbol{y})} = \boldsymbol{x} - \dfrac{2(\boldsymbol{x}-\boldsymbol{y})(\boldsymbol{x}^t\boldsymbol{x}-\boldsymbol{y}^t\boldsymbol{x})}{\boldsymbol{x}^t\boldsymbol{x}-\boldsymbol{y}^t\boldsymbol{x}-\boldsymbol{x}^t\boldsymbol{y}+\boldsymbol{y}^t\boldsymbol{y}}$ $= \boldsymbol{x} - \dfrac{2(\boldsymbol{x}^t\boldsymbol{x}-\boldsymbol{y}^t\boldsymbol{x})}{2(\boldsymbol{x}^t\boldsymbol{x}-\boldsymbol{y}^t\boldsymbol{x})}(\boldsymbol{x}-\boldsymbol{y}) = \boldsymbol{y}$
p.192, 定理 9.4 の証明の最後から 2 行目	なので，第 k 行までヘッセンベルグ行列になります.	なので，第 k 列までヘッセンベルグ行列になります.
p.192，(9.24) の 2 行目と 3 行目	$A_{k+1} = P_kA_kP_k = (E_n - 2\boldsymbol{u}_k\boldsymbol{u}_k^t)A_k(E_n - 2\boldsymbol{u}_k\boldsymbol{u}_k^t)$ $= A_k - 2A\boldsymbol{u}_k\boldsymbol{u}_k^t - 2\boldsymbol{u}_k\boldsymbol{u}_k^tA_k + 4\boldsymbol{u}_k\boldsymbol{u}_k^tA_k\boldsymbol{u}_k\boldsymbol{u}_k^t$ $= A_k - 2\boldsymbol{u}_k\{\boldsymbol{u}_k^tA_k - (\boldsymbol{u}_k^tA_k\boldsymbol{u}_k)\boldsymbol{u}_k^t\} - 2\{A\boldsymbol{u}_k - \boldsymbol{u}_k(\boldsymbol{u}_k^tA_k\boldsymbol{u}_k)\}\boldsymbol{u}_k^t$	$A_{k+1} = P_kA_kP_k = (E_n - 2\boldsymbol{u}_k\boldsymbol{u}_k^t)A_k(E_n - 2\boldsymbol{u}_k\boldsymbol{u}_k^t)$ $= A_k - 2A_k\boldsymbol{u}_k\boldsymbol{u}_k^t - 2\boldsymbol{u}_k\boldsymbol{u}_k^tA_k + 4\boldsymbol{u}_k\boldsymbol{u}_k^tA_k\boldsymbol{u}_k\boldsymbol{u}_k^t$ $= A_k - 2\boldsymbol{u}_k\{\boldsymbol{u}_k^tA_k - (\boldsymbol{u}_k^tA_k\boldsymbol{u}_k)\boldsymbol{u}_k^t\} - 2\{A_k\boldsymbol{u}_k - \boldsymbol{u}_k(\boldsymbol{u}_k^tA_k\boldsymbol{u}_k)\}\boldsymbol{u}_k^t$
p.200，ページ最後の行列	$P_2P_1A = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & \cdots & \cdots & r_{1n} \\ 0 & r_{21} & r_{22} & \cdots & \cdots & \cdots & r_{2n} \\ 0 & 0 & a'_{33} & \cdots & \cdots & \cdots & a'_{3n} \\ 0 & 0 & a_{43} & \cdots & \cdots & \cdots & a_{4n} \\ \vdots & \vdots & 0 & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a_{n,n-1} & a_{nn} \end{bmatrix}$	$P_2P_1A = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & \cdots & \cdots & r_{1n} \\ 0 & r_{22} & r_{23} & \cdots & \cdots & \cdots & r_{2n} \\ 0 & 0 & a'_{33} & \cdots & \cdots & \cdots & a'_{3n} \\ 0 & 0 & a_{43} & \cdots & \cdots & \cdots & a_{4n} \\ \vdots & \vdots & 0 & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a_{n,n-1} & a_{nn} \end{bmatrix}$

	誤	正
p.202，QR 法のアルゴリズムの前	ここでは，A_k の n 行 n 列成分を順次 s_k と選ぶことにします．	ここでは，A_k の n 行 n 列成分を s_k と選ぶことにします．なお，$m = n$ のときは，1 つも近似固有値が求まっていないので，このような原点移動はできないことに注意してください．
p.202，QR 法のアルゴリズム	`if` $\lvert a_{m,m-1} < \varepsilon \rvert$ `then`	`if` $\lvert a_{m,m-1} \rvert < \varepsilon$ `then`
p.202，QR 法のアルゴリズム	`/* 原点移動 */`	`/* 原点移動（ただし，`$m = n$`のときは除く） */`
p.204 プログラム 9.3 の QR 法の関数内	`while ( m >= 1)`	`while ( m > 1)`

p.204，プログラム中の原点移動	`/* 原点移動 */`
	`s = a[n][n];`
	`for ( i = 1; i <= m; i++) a[i][i] -= s ;`
		`/* 原点移動 */`
		`s = a[n][n];`
		`if ( m == n ) s = 0.0; /* m=n のときは原点移動なし */`
		`for ( i = 1; i <= m; i++) a[i][i] -= s ;`

p.210，逆反復法の項，下線部を削除	ここまでの話で，QR 法を使うと行列 A の近似固有値を求められることが分かりました．ここでは，行列 A の近似固有値 $\hat{\lambda}_i$ の固有値が与えられたとき，A の固有値 λ_i に対する固有ベクトル $\boldsymbol{x}_i$ を求めることを考えます．	ここまでの話で，QR 法を使うと行列 A の近似固有値を求められることが分かりました．ここでは，行列 A の近似固有値 $\hat{\lambda}_i$ が与えられたとき，A の固有値 λ_i に対する固有ベクトル $\boldsymbol{x}_i$ を求めることを考えます．
p.210，中程	を満たすと仮定すれば，$\frac{1}{\lambda_i - \hat{\lambda}_i}$ は $(A - \hat{\lambda}_i E_n)$ の絶対値最大の固有値となります．	を満たすと仮定すれば，$\frac{1}{\lambda_i - \hat{\lambda}_i}$ は $(A - \hat{\lambda}_i E_n)^{-1}$ の絶対値最大の固有値となります．
p.211，2 行目	また，反復の各段階で (9.10) と同様に考えれば，	また，反復の各段階で (9.11) と同様に考えれば，
p.211，逆反復法のアルゴリズム	Input A，λ_i，n，ε ⋮ `while` $\lVert \boldsymbol{v} \rVert_2^2 - \lvert \mu_i^{(k)} \rvert^2 < \varepsilon$	Input A，$\hat{\lambda}_i$，n，ε ⋮ `while` $(\mu_i^{(k)} - \mu_i^{(k-1)})/\mu_i^{(k)} \geq \varepsilon$
p.211，逆反復法のアルゴリズムの直後	以下に，逆反復法のプログラム例を示します．	このアルゴリズムでは，べき乗法のアルゴリズムと異なり，収束判定条件を $(\mu_i^{(k)} - \mu_i^{(k-1)})/\mu_i^{(k)} < \varepsilon$ としています．というのも，$(A - \hat{\lambda}_i E_n)$ はほぼ非正則行列と考えられ，そのため $\boldsymbol{v}$ が精度良く求まらない可能性があるためです．このようなときは，単純に考えて，反復により $\mu_i^{(k)}$ や $\lVert \boldsymbol{y}^{(k)} \rVert$ の値があまり変化しなくなったら反復を打ち切る，とした方が無難です． 以下に，逆反復法のプログラム例を示します．

<table>
<tr><th></th><th>誤</th><th>正</th></tr>
<tr><td>p.213 プログラム 9.4 の逆反復法の関数内</td><td>

```
double v2, v2s, *v, *y, **lu, lambda, mu;
            ⋮

    /* 逆反復法 */
    do
    {
      v = lu_solve( lu, y, p );  /* 固有ベクトルの計算 */
      mu = inner_product( 1, N, v, y );  /* 補正 */
      v2 = inner_product( 1, N, v, v );
      v2s = sqrt(v2);
      for( j = 1; j <= N ; j++) y[j]=v[j]/v2s;
    }while( fabs(v2-mu) >= eps );
```

</td><td>

```
double v2, v2s, *v, *y, **lu, lambda, mu=0.0, muo;
          ⋮
    /* 逆反復法 */
    do
    {
      muo = mu;
      for ( j = 1; j <= N; j++ ) v[j] = y[j];
      v = lu_solve( lu, v, p ); /* 固有ベクトルの計算 */
      mu = inner_product( 1, N, v, y );  /* 補正 */
      v2 = inner_product( 1, N, v, v );
      v2s = sqrt(v2);
      for( j = 1; j <= N ; j++) y[j]=v[j]/v2s;
}while( fabs((mu-muo)/mu) >= eps );
```

</td></tr>
<tr><td>p.214 の実行例, 固有ベクトルの符号を逆にする</td><td>

```
固有値は
-24.1485206     21.5793450      14.3382192      12.2309564
固有ベクトルは
[ 0.0244773      0.0268830      -0.9991047       0.0216307      ]
[ 0.3477018     -0.0201900       0.0473792       0.9361896      ]
[ 0.6705354      0.7004858       0.0599467      -0.2368719      ]
[-0.1466821     -0.9712629      -0.0934406      -0.1624855      ]
```

</td><td>

```
固有値は
-24.1485206     21.5793450      14.3382192      12.2309564
固有ベクトルは

[-0.0244773    -0.0268830     0.9991047    -0.0216307     ]
[-0.3477018     0.0201900    -0.0473792    -0.9361896     ]
[-0.6705354    -0.7004858    -0.0599467     0.2368719     ]
[ 0.1466821     0.9712629     0.0934406     0.1624855     ]
```

</td></tr>
<tr><td>p.225 の main 関数から変数 i を削除</td><td>

```
int a[N+1], b[N+1], pi[N+1], i;
```

</td><td>

```
int a[N+1], b[N+1], pi[N+1];
```

</td></tr>
</table>