システム工学概論

5. Cによる数値計算と可視化（応用編）

山田泰司

taiji@aihara.co.jp

株式会社あいはら 研究開発チーム

（例題１）数値解析と３次元グラフィックス

以下のローレンツ方程式と呼ばれる常微分方程式：

$$\begin{aligned}\frac{dx}{dt} &= -\sigma x + \sigma y \\ \frac{dy}{dt} &= -xz + rx - y \\ \frac{dz}{dt} &= xy - bz\end{aligned}$$

において $\sigma = 10, b = 8/3, r = 28, x_0 = 10, y_0 = 20, z_0 = 30$ **における** $t < 10000\delta t$ **(例えば、** $\delta t = 0.01$**) までの値をディスプレイにて可視化せよ。**

数値解析の手段

- **アルゴリズム・数値解析に関する書籍や論文等を参考に（自前で）実装する**
  ☞ **例えば、奥村 著「C言語による最新アルゴリズム事典」技術評論社**, 1991.
- **書籍「**Numerical Recipes in C**」を参考にする**
- Netlib **の** LAPACK, BLAS **等を利用する**
  ☞ `http://www.netlib.org/`
- gsl(GNU Scientific Library) **を活用する**
  ☞ `http://www.gnu.org/software/gsl/`
- **その他、手法を自分で研究し、実装する**
- **もしくは、**Matlab, Octave, Scilab **等、数値計算パッケージソフトを利用する**

３次元グラフィックスの手段

- X11 **等の２次元座標系へ（自前で）３次元空間を投影する**
- OpenGL **等の３次元グラフィックス技術を利用する**

OpenGL **プログラミングの選択肢**

- GLUT — The OpenGL Utility Toolkit
- OpenGL + X11 — glx
- OpenGL + SDL
- **その他、各プラットフォームのウィンドウィングに沿った形態**

OpenGL **以外の選択肢：**

- MESA — **マルチプラットフォーム** OpenGL **互換ライブラリ**
- DirectX — Windows **用**
- xgl(X11 over OpenGL) — **現在、**Linux **用**
- **その他、**POV-Ray **などのレンダラ**

数値計算と３次元グラフィックス

- ▷ 簡単な３次元グラフィックスなら自前で用意することはそれほど難しくない
  - ⇨ 回転変換、四元数、透視変換など
- ▷ しかし、とことんリアリティを求めるなら OpenGL 等を検討すべき
  - ⇨ ポリゴン、曲面
  - ⇨ レイトレーシング
  - ⇨ テクスチャマッピング、シェーディングなど

その他、イベント処理等のウィンドウィング処理は OpenGL 以外の技術を要する。

３次元グラフィックスの学び方

- E. Lengyel(**狩野 訳**)**「ゲームプログラミングのための** 3D **グラフィックス数学」株式会社ボーンデジタル**, 2002.
- D. F. Rogers & J. A. Adams(**川合 他 訳**)**「コンピュータグラフィックス」日刊工業新聞社**, 1993.
- OpenGL **策定委員会** (**松田 訳**)**「**OpenGL **プログラミングガイド 第** 5 **版」ピアソンエデュケーション**, 2006.
- glx **のマニュアル**

  ```
  $ man glXIntro
  ```

- SDL **のドキュメントやサンプルコード**
  `http://www.libsdl.org/` **の** OpenGL **関係など**

（例題１）の解答例0-1/4

常微分方程式の求解法のひとつ — ４次のルンゲ・クッタ法：

```
          :
void ode_rk4(double p[], double delta_t, int N,
             void (*dvdt)(double p[], double t, double v[], double dvdt[]),
             double t, double v0[], double v1[])
{
  int i;
  double dv[N], d1[N], d2[N], d3[N], va[N];

  dvdt(p, t, v0, dv);
  for (i=0; i<N; i++) {
    d1[i] = delta_t*dv[i];
    va[i] = v0[i] + 0.5*d1[i];
  }
  dvdt(p, t + 0.5*delta_t, va, dv);
  for (i=0; i<N; i++) {
    d2[i] = delta_t*dv[i];
    va[i] = v0[i] + 0.5*d2[i];
  }
  dvdt(p, t + 0.5*delta_t, va, dv);
  for (i=0; i<N; i++) {
    d3[i] = delta_t*dv[i];
    va[i] = v0[i] + d3[i];
  }
  dvdt(p, t + delta_t, va, dv);
  for (i=0; i<N; i++)
    v1[i] = v0[i] + (d1[i]+d2[i]*2+d3[i]*2+delta_t*dv[i])/6;
}
```

（例題１）の解答例0-2/4

構造体 ode、ローレンツ方程式の初期化、オプション設定、微分方程式の準備：

```
struct ode {
  int K, N;
  void (*init)(double p[], double v0[]);
  void (*options)(int argc, char *argv[], double p[], double v0[]);
  void (*dvdt)(double p[], double t, double v[], double dvdt[]);
};
void lorenz_init(double p[], double v[])
{
#define sigma p[0]
#define b p[1]
#define r p[2]
#define x v[0]
#define y v[1]
#define z v[2]
  sigma = 10.0; b = 8.0/3; r = 28.0; x = 10.0; y = 20.0; z = 30.0;
}
void lorenz_dvdt(double p[], double t, double v[], double dvdt[])
{
  dvdt[0] = -sigma*x + sigma*y;
  dvdt[1] = -x*z + r*x - y;
  dvdt[2] = x*y - b*z;
}
void lorenz_options(int argc, char *argv[], double p[], double v[])
        :
struct ode lorenz_ode = {
  3, 3, lorenz_init, lorenz_options, lorenz_dvdt
}, *ode = &lorenz_ode;
```

（例題１）の解答例0-3/4

ode と常微分方程式の求解の時間幅、パラメータ、状態変数等からなる構造体 odeset の準備、及びその出力ルーチン：

```
struct odeset {
  struct ode *ode;
  double delta_t, *p, *v0, *v1;
};
void odes_write(struct odeset *odes)
{
  int i, j;

  for (i=0; i<10000; i++) {
    ode_rk4(odes->p, odes->delta_t, odes->ode->N, odes->ode->dvdt,
            odes->delta_t*i, odes->v0, odes->v1);
    for (j=0; j<odes->ode->N; j++)
      printf("%g\t", odes->v0[j]);
    printf("\n");
    for (j=0; j<odes->ode->N; j++)
      odes->v0[j] = odes->v1[j];
  }
}
```

とりあえずは、数値計算結果を出力するところから始める。

（例題１）の解答例0-4/4

構造体 `odeset` **の実体を用意して、出力ルーチンの呼び出し：**

```
int main(int argc, char *argv[])
{
  struct odeset odeset = {
    ode,
    0.01,
    malloc(sizeof(double)*ode->K),
    malloc(sizeof(double)*ode->N),
    malloc(sizeof(double)*ode->N),
  }, *odes = &odeset;

  odes->ode->init(odes->p, odes->v0);
  odes->ode->options(argc, argv, odes->p, odes->v0);
  odes_write(odes);
  return 0;
}
```

```
$ ./v_lorenz00 > lorenz.dat
$ echo 'splot "lorenz.dat" notitle with lines; pause mouse;' | gnuplot -
```

（例題１）の解答例1-1/6

$ diff -u v_lorenz00.c v_lorenz01.c # **とりあえず、２次元グラフィックスから**

```
 @@ -1,6 +1,7 @@
  #include <stdio.h>     /* printf */
  #include <stdlib.h>    /* atof */
  #include <string.h>    /* strcmp */
 +#include <X11/Xlib.h>
 @@ -35,14 +36,14 @@
   */
  struct ode {
    int K, N;
 -  void (*init)(double p[], double v0[]);
 +  void (*init)(double p[], double v0[], double lb[], double ub[]);
    void (*options)(int argc, char *argv[], double p[], double v0[]);
    void (*dvdt)(double p[], double t, double v[], double dvdt[]);
  };
 -void lorenz_init(double p[], double v[])
 +void lorenz_init(double p[], double v[], double lb[], double ub[])
  {
  #define sigma p[0]
  #define b p[1]
```

（例題１）の解答例1-2/6

```
@@ -50,12 +51,21 @@
 #define x v[0]
 #define y v[1]
 #define z v[2]
-  sigma = 10.0; b = 8.0/3; r = 28.0; x = 10.0; y = 20.0; z = 30.0;
+  if (p) {
+    sigma = 10.0; b = 8.0/3; r = 28.0;
+  }
+  if (v) {
+    x = 10.0; y = 20.0; z = 30.0;
+  }
+  if (lb && ub) {
+    lb[0] = -20; ub[0] = 20;
+    lb[1] = -30; ub[1] = 30;
+    lb[2] = 5; ub[2] = 50;
+  }
 }
 void lorenz_dvdt(double p[], double t, double v[], double dvdt[])
 {
@@ -96,20 +106,97 @@
 struct odeset {
   struct ode *ode;
   double delta_t, *p, *v0, *v1;
+  double *lb, *ub;
+};
```

（例題１）の解答例1-3/6

```
+struct view {
+  int w, h;
+  Display *display; int screen_number; Window parent_window, window; Pixmap pixmap;
+  GC gc; Colormap cmap; long event_mask; XEvent event;
 };
-void odes_write(struct odeset *odes)
+void view_init(struct view *v)
+{
+  if (!(v->display = XOpenDisplay(NULL)))
+    exit(1);
+  v->screen_number = DefaultScreen(v->display);
+  v->parent_window = RootWindow(v->display, v->screen_number);
+  v->window = XCreateSimpleWindow(v->display, v->parent_window,
+                                  0, 0, v->w, v->h, 0,
+                                  BlackPixel(v->display, v->screen_number),
+                                  WhitePixel(v->display, v->screen_number));
+  v->pixmap = XCreatePixmap(v->display, v->window, v->w, v->h,
+                            DefaultDepth(v->display, v->screen_number));
+  v->gc = DefaultGC(v->display, v->screen_number);
+  v->cmap = DefaultColormap(v->display, v->screen_number);
+  XSetForeground(v->display, v->gc, BlackPixel(v->display, v->screen_number));
+  v->event_mask = ExposureMask;
+  v->event_mask |= KeyPressMask;
+  v->event_mask |= StructureNotifyMask;
+  XSelectInput(v->display, v->window, v->event_mask);
+  XMapWindow(v->display, v->window);
+  XFlush(v->display);
+}
```

（例題１）の解答例1-4/6

```
+void view_draw(struct view *v, void *o)
 {
+  struct odeset *odes = (struct odeset *)o;
   int i, j;
+  int x0, y0, x1, y1;
+  XSetForeground(v->display, v->gc, WhitePixel(v->display, v->screen_number));
+  XFillRectangle(v->display, v->pixmap, v->gc, 0, 0, v->w, v->h);
+  XSetForeground(v->display, v->gc, BlackPixel(v->display, v->screen_number));
+  odes->ode->init(NULL, odes->v0, NULL, NULL);
   for (i=0; i<10000; i++) {
     ode_rk4(odes->p, odes->delta_t, odes->ode->N, odes->ode->dvdt,
             odes->delta_t*i, odes->v0, odes->v1);
-    for (j=0; j<odes->ode->N; j++) printf("%g\t", odes->v0[j]); printf("\n");
+    if (i == 0) {
+      x0 = (odes->v0[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0]) * v->w;
+      y0 = (odes->v0[1] - odes->ub[1])/(odes->lb[1] - odes->ub[1]) * v->h;
+    }
+    else {
+      x0 = x1;
+      y0 = y1;
+    }
+    x1 = (odes->v1[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0]) * v->w;
+    y1 = (odes->v1[1] - odes->ub[1])/(odes->lb[1] - odes->ub[1]) * v->h;
+    XDrawLine(v->display, v->pixmap, v->gc, x0, y0, x1, y1);
     for (j=0; j<odes->ode->N; j++) odes->v0[j] = odes->v1[j];
   }
+  XCopyArea(v->display, v->pixmap, v->window, v->gc, 0, 0, v->w, v->h, 0, 0);
+  XFlush(v->display);
+}
```

（例題１）の解答例1-5/6

```
+void view_loop(struct view *v, void *o)
+{
+  while (!0) {
+    XNextEvent(v->display, &v->event);
+    switch (v->event.type) {
+    case Expose: view_draw(v, o); break;
+    case KeyPress: return; break;
+    case ConfigureNotify:
+      v->w = v->event.xconfigure.width;
+      v->h = v->event.xconfigure.height;
+      XFreePixmap(v->display, v->pixmap);
+      v->pixmap = XCreatePixmap(v->display, v->window, v->w, v->h,
+                                DefaultDepth(v->display, v->screen_number));
+      view_draw(v, o);
+      break;
+    }
+  }
+}
+void view_term(struct view *v)
+{
+  XUnmapWindow(v->display, v->window);
+  XCloseDisplay(v->display);
 }
```

（例題１）の解答例1-6/6

```
 int main(int argc, char *argv[])
@@ -120,10 +207,17 @@
     malloc(sizeof(double)*ode->K),
     malloc(sizeof(double)*ode->N),
     malloc(sizeof(double)*ode->N),
+    malloc(sizeof(double)*ode->N),
+    malloc(sizeof(double)*ode->N),
   }, *odes = &odeset;
+  struct view view = {
+    600, 600,
+  }, *v = &view;

-  odes->ode->init(odes->p, odes->v0);
+  odes->ode->init(odes->p, odes->v0, odes->lb, odes->ub);
   odes->ode->options(argc, argv, odes->p, odes->v0);
-  odes_write(odes);
+  view_init(v);
+  view_loop(v, odes);
+  view_term(v);
   return 0;
 }
```

```
$ ./v_lorenz01.c
```

（例題１）の解答例2-1/1

```
$ diff -u v_lorenz01.c v_lorenz02.c # 画面に文字を書く
```

```
@@ -154,6 +154,13 @@
   XFillRectangle(v->display, v->pixmap, v->gc, 0, 0, v->w, v->h);
   XSetForeground(v->display, v->gc, BlackPixel(v->display, v->screen_number));
   odes->ode->init(NULL, odes->v0, NULL, NULL);
+  {
+    char str[256];
+
+    snprintf(str, sizeof(str)/sizeof(char), "sigma=%g,b=%g,r=%g",
+            odes->p[0], odes->p[1], odes->p[2]);
+    XDrawString(v->display, v->pixmap, v->gc, 8, 16, str, strlen(str));
+  }
   for (i=0; i<10000; i++) {
     ode_rk4(odes->p, odes->delta_t, odes->ode->N, odes->ode->dvdt,
            odes->delta_t*i, odes->v0, odes->v1);
```

```
$ ./v_lorenz02.c
```

（例題１）の解答例3-1/4

$ diff -u v_lorenz02.c v_lorenz03.c # ３次元グラフィックスを試みる

```
          :
 +double deg2rad(double deg)
 +{
 +  return deg*M_PI/180;
 +}
 +void xrot_mat3(double a, double m[3][3])
 +{
 +  double c = cos(a), s = sin(a);
 +  m[0][0] = 1; m[0][1] = 0; m[0][2] =  0;
 +  m[1][0] = 0; m[1][1] = c; m[1][2] = -s;
 +  m[2][0] = 0; m[2][1] = s; m[2][2] =  c;
 +}
 +void yrot_mat3(double a, double m[3][3])
 +{
 +  double c = cos(a), s = sin(a);
 +  m[0][0] =  c; m[0][1] = 0; m[0][2] = s;
 +  m[1][0] =  0; m[1][1] = 1; m[1][2] = 0;
 +  m[2][0] = -s; m[2][1] = 0; m[2][2] = c;
 +}
 +void zrot_mat3(double a, double m[3][3])
 +{
 +  double c = cos(a), s = sin(a);
 +  m[0][0] = c; m[0][1] = -s; m[0][2] = 0;
 +  m[1][0] = s; m[1][1] =  c; m[1][2] = 0;
 +  m[2][0] = 0; m[2][1] =  0; m[2][2] = 1;
 +}
```

（例題１）の解答例3-2/4

```
+void mat3_mult_mat3(double a[3][3], double b[3][3], double c[3][3])
+{
+  int i, j, k;
+
+  for (i=0; i<3; i++)
+    for (j=0; j<3; j++)
+      for (c[i][j] = 0, k=0; k<3; k++)
+        c[i][j] += a[i][k]*b[k][j];
+}
+void mat3_mult_vec3(double a[3][3], double b[3], double c[3])
+{
+  int i, j;
+
+  for (i=0; i<3; i++)
+    for (c[i] = 0, j=0; j<3; j++)
+      c[i] += a[i][j]*b[j];
+}
 struct view {
    int w, h;
+  double deg[3], s;
   Display *display;
   int screen_number;
   Window parent_window, window;
```

（例題１）の解答例3-3/4

```
@@ -149,6 +199,7 @@
   struct odeset *odes = (struct odeset *)o;
   int i, j;
   int x0, y0, x1, y1;
+  double mat[3][3];

   XSetForeground(v->display, v->gc, WhitePixel(v->display, v->screen_number));
   XFillRectangle(v->display, v->pixmap, v->gc, 0, 0, v->w, v->h);
@@ -161,19 +212,36 @@
             odes->p[0], odes->p[1], odes->p[2]);
     XDrawString(v->display, v->pixmap, v->gc, 8, 16, str, strlen(str));
   }
+  {
+    double matx[3][3], maty[3][3], matz[3][3], matw[3][3];
+
+    xrot_mat3(deg2rad(v->deg[0]), matx);
+    yrot_mat3(deg2rad(v->deg[1]), maty);
+    zrot_mat3(deg2rad(v->deg[2]), matz);
+    mat3_mult_mat3(matx, maty, matw);
+    mat3_mult_mat3(matw, matz, mat);
+  }
```

（例題１）の解答例3-4/4

```
    for (i=0; i<10000; i++) {
+     double p[3], r[3];
      ode_rk4(odes->p, odes->delta_t, odes->ode->N, odes->ode->dvdt,
              odes->delta_t*i, odes->v0, odes->v1);
      if (i == 0) {
-       x0 = (odes->v0[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0]) * v->w;
-       y0 = (odes->v0[1] - odes->ub[1])/(odes->lb[1] - odes->ub[1]) * v->h;
+       p[0] = (odes->v0[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0])* 2 - 1;
+       p[1] = (odes->v0[1] - odes->lb[1])/(odes->ub[1] - odes->lb[1])* 2 - 1;
+       p[2] = (odes->v0[2] - odes->lb[2])/(odes->ub[2] - odes->lb[2])* 2 - 1;
+       mat3_mult_vec3(mat, p, r);
+       x0 =  (r[0]*v->s + 1)*v->w/2; y0 = (-r[1]*v->s + 1)*v->h/2;
      }
         :
      {
-       x1 = (odes->v1[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0]) * v->w;
-       y1 = (odes->v1[1] - odes->ub[1])/(odes->lb[1] - odes->ub[1]) * v->h;
+       p[0] = (odes->v1[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0])* 2 - 1;
+       p[1] = (odes->v1[1] - odes->lb[1])/(odes->ub[1] - odes->lb[1])* 2 - 1;
+       p[2] = (odes->v1[2] - odes->lb[2])/(odes->ub[2] - odes->lb[2])* 2 - 1;
+       mat3_mult_vec3(mat, p, r);
+       x1 =  (r[0]*v->s + 1)*v->w/2; y1 = (-r[1]*v->s + 1)*v->h/2;
      }
      XDrawLine(v->display, v->pixmap, v->gc, x0, y0, x1, y1);
```

```
$ ./v_lorenz03.c
```

（例題１）の解答例4-1/3

$ diff -u v_lorenz03.c v_lorenz04.c # **色空間も活用する**

```
+double vec_dist(int N, double a[], double b[])
+{
+  int i;
+  double d = 0;
+
+  for (i=0; i<N; i++)
+    d += (a[i] - b[i])*(a[i] - b[i]);
+  return sqrt(d);
+}
        :
+void HSV2RGB(double h, double s, double v, double *r, double *g, double *b)
+{
+  double R, G, B, f, i, m, n, k;
+
+  h *= 6; f = modf(h, &i);
+  m = v*(1 - s); n = v*(1 - s*f); k = v*(1 - s*(1 - f));
+  switch ((int)i) {
+  case 0: R = v; G = k; B = m; break;
+  case 1: R = n; G = v; B = m; break;
+  case 2: R = m; G = v; B = k; break;
+  case 3: R = m; G = n; B = v; break;
+  case 4: R = k; G = m; B = v; break;
+  case 5: R = v; G = m; B = n; break;
+  }
+  *r = R; *g = G; *b = B;
+}
```

（例題１）の解答例4-2/3

```
          :
+     lb[3] = 0; ub[3] = 400;
          :
@@ -169,6 +207,7 @@
    Pixmap pixmap;
    GC gc;
    Colormap cmap;
+   XColor colors[1024];
    long event_mask;
    XEvent event;
  };
@@ -186,6 +225,19 @@
                                DefaultDepth(v->display, v->screen_number));
    v->gc = DefaultGC(v->display, v->screen_number);
    v->cmap = DefaultColormap(v->display, v->screen_number);
+   {
+     int c, ncolors = sizeof(v->colors)/sizeof(XColor);
+
+     for (c=0; c<ncolors; c++) {
+       double R, G, B;
+
+       HSV2RGB((double)c/ncolors, 1, 1, &R, &G, &B);
+       v->colors[c].red = 65535*R;
+       v->colors[c].green = 65535*G;
+       v->colors[c].blue = 65535*B;
+       XAllocColor(v->display, v->cmap, &v->colors[c]);
+     }
+   }
```

（例題１）の解答例4-3/3

```
          :
-   int x0, y0, x1, y1;
-   double mat[3][3];
+   int x0, y0, x1, y1, c, ncolors = sizeof(v->colors)/sizeof(XColor);
+   double velocity01, mat[3][3];

    XSetForeground(v->display, v->gc, WhitePixel(v->display, v->screen_number));
    XFillRectangle(v->display, v->pixmap, v->gc, 0, 0, v->w, v->h);
:
      ode_rk4(odes->p, odes->delta_t, odes->ode->N, odes->ode->dvdt,
              odes->delta_t*i, odes->v0, odes->v1);
+     velocity01 = vec_dist(odes->ode->N, odes->v0, odes->v1)/odes->delta_t;
      if (i == 0) {
        p[0] = (odes->v0[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0])* 2 - 1;
        p[1] = (odes->v0[1] - odes->lb[1])/(odes->ub[1] - odes->lb[1])* 2 - 1;
:
        mat3_mult_vec3(mat, p, r);
        x1 =  (r[0]*v->s + 1)*v->w/2; y1 = (-r[1]*v->s + 1)*v->h/2;
      }
+     c = (velocity01 - odes->lb[3])/(odes->ub[3] - odes->lb[3]) * ncolors;
+     XSetForeground(v->display, v->gc, (0<=c&&c<ncolors)?
+                    v->colors[c].pixel:BlackPixel(v->display, v->screen_number));
      XDrawLine(v->display, v->pixmap, v->gc, x0, y0, x1, y1);
          :
```

```
$ ./v_lorenz04.c
```

（例題１）の解答例5-1/2

$ diff -u v_lorenz04.c v_lorenz05.c # **マウスで回転変換の角度を変える**

```
 struct view {
   int w, h;
   double deg[3], s;
+  int pointer[2];
   Display *display;
   int screen_number;
   Window parent_window, window;
@@ -241,6 +242,7 @@
   XSetForeground(v->display, v->gc, BlackPixel(v->display, v->screen_number));
   v->event_mask = ExposureMask;
   v->event_mask |= KeyPressMask;
+  v->event_mask |= ButtonPressMask|ButtonMotionMask;
   v->event_mask |= StructureNotifyMask;
   XSelectInput(v->display, v->window, v->event_mask);
   XMapWindow(v->display, v->window);
```

（例題１）の解答例5-2/2

```
@@ -312,6 +314,17 @@
     XNextEvent(v->display, &v->event);
     switch (v->event.type) {
     case Expose: view_draw(v, o); break;
+    case ButtonPress:
+      v->pointer[0] = v->event.xbutton.x;
+      v->pointer[1] = v->event.xbutton.y;
+      break;
+    case MotionNotify:
+      v->deg[0] += (v->event.xmotion.y - v->pointer[1])*180/v->h;
+      v->deg[2] += (v->event.xmotion.x - v->pointer[0])*180/v->w;
+      v->pointer[0] = v->event.xmotion.x;
+      v->pointer[1] = v->event.xmotion.y;
+      view_draw(v, o);
+      break;
     case KeyPress: return; break;
     case ConfigureNotify:
       v->w = v->event.xconfigure.width;
```

```
$ ./v_lorenz05.c
```

（例題１）の解答例6-1/3

$ diff -u v_lorenz05.c v_lorenz.c # **回転変換行列の代わりに四元数を使う**

```
+void co_qua(double q[4], double qo[4])
        :
+void qua_mult_qua(double q1[4], double q2[4], double qo[4])
        :
+void qua_mult_vec3(double q[4], double v[3], double qo[4])
        :
+void ang_to_qua(double a[3], double q[4])
        :
+void qua_rot_vec3(double q[4], double v[3], double qo[4])
+{
+  double qv[4], coq[4];
+
+  qua_mult_vec3(q, v, qv);
+  co_qua(q, coq);
+  qua_mult_qua(qv, coq, qo);
+}
+void mat3_to_ang(double m[3][3], double a[3])
        :
```

（例題１）の解答例6-2/3

```
    struct odeset *odes = (struct odeset *)o;
    int i, j;
    int x0, y0, x1, y1, c;
-   double velocity01, mat[3][3];
+   double velocity01, mat[3][3], qua[4];

    XSetForeground(v->display, v->gc, WhitePixel(v->display, v->screen_number));
    XFillRectangle(v->display, v->pixmap, v->gc, 0, 0, v->w, v->h);
@@ -267,13 +364,15 @@
      XDrawString(v->display, v->pixmap, v->gc, 8, 16, str, strlen(str));
    }
    {
-     double matx[3][3], maty[3][3], matz[3][3], matw[3][3];
+     double matx[3][3], maty[3][3], matz[3][3], matw[3][3], ang[3];

      xrot_mat3(deg2rad(v->deg[0]), matx);
      yrot_mat3(deg2rad(v->deg[1]), maty);
      zrot_mat3(deg2rad(v->deg[2]), matz);
      mat3_mult_mat3(matx, maty, matw);
      mat3_mult_mat3(matw, matz, mat);
+     mat3_to_ang(mat, ang);
+     ang_to_qua(ang, qua);
    }
```

（例題１）の解答例6-3/3

```
    for (i=0; i<10000; i++) {
-     double p[3], r[3];
+     double p[3], r[4];

      ode_rk4(odes->p, odes->delta_t, odes->ode->N, odes->ode->dvdt,
              odes->delta_t*i, odes->v0, odes->v1);
@@ -285,7 +384,7 @@
        p[0] = (odes->v0[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0])* 2 - 1;
        p[1] = (odes->v0[1] - odes->lb[1])/(odes->ub[1] - odes->lb[1])* 2 - 1;
        p[2] = (odes->v0[2] - odes->lb[2])/(odes->ub[2] - odes->lb[2])* 2 - 1;
-       mat3_mult_vec3(mat, p, r);
+       qua_rot_vec3(qua, p, r);
        x0 =  (r[0]*v->s + 1)*v->w/2; y0 = (-r[1]*v->s + 1)*v->h/2;
      }
      else {
@@ -295,7 +394,7 @@
        p[0] = (odes->v1[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0])* 2 - 1;
        p[1] = (odes->v1[1] - odes->lb[1])/(odes->ub[1] - odes->lb[1])* 2 - 1;
        p[2] = (odes->v1[2] - odes->lb[2])/(odes->ub[2] - odes->lb[2])* 2 - 1;
-       mat3_mult_vec3(mat, p, r);
+       qua_rot_vec3(qua, p, r);
        x1 =  (r[0]*v->s + 1)*v->w/2; y1 = (-r[1]*v->s + 1)*v->h/2;
      }
      c = (velocity01 - odes->lb[3])/(odes->ub[3] - odes->lb[3]) * ncolors;
```

```
$ ./v_lorenz.c
```

gsl(GNU Scientific Library)の利用

「gsl がカバーする分野」
数学関数、複素数、多項式、特殊関数、ベクトルと行列演算、置換、組み合わせ、整列、線形代数、高速な基本線形代数 (BLAS: The Basic Linear Algebra Subprograms)**、固有値問題、フーリエ変換、数値積分、乱数・乱数列、確率分布、統計、N 個組、モンテカルロ積分、アニーリング、常微分方程式、補間、数値微分、チェビシェフ近似、級数、ウェーブレット、離散ハンケル変換、一次元・多次元関数の求根・最適化、線形・非線形最小二乗法、物理定数、**IEEE **浮動小数点**

『gsl のインストール』

```
$ wget -N ftp://ftp.gnu.org/gnu/gsl/gsl-1.8.tar.gz
$ tar tvzf gsl-1.8.tar.gz | less
$ tar xvzf gsl-1.8.tar.gz
$ cd gsl-1.8
$ ./configure --help | less
$ ./configure --prefix=$HOME/local
$ make
$ make install
```

『gsl のアンインストール』

```
$ make uninstall
$ make -n uninstall | less
```

（例題１）の解答例7-1/2

```
$ cd ../x11+gsl
$ diff -u ../x11/v_lorenz.c v_lorenz.c # gsl を利用する
```

```
 #include <string.h>    /* strcmp */
 #include <math.h>      /* sin, cos, sqrt, modf */
 #include <X11/Xlib.h>
+#include <gsl/gsl_matrix.h>
+#include <gsl/gsl_odeiv.h>
        :
 struct ode {
   int K, N;
   void (*init)(double p[], double v0[], double lb[], double ub[]);
   void (*options)(int argc, char *argv[], double p[], double v0[]);
   void (*dvdt)(double p[], double t, double v[], double dvdt[]);
+  int (*dvdt4gsl)(double t, const double v[], double dvdt[], void *p);
 };
        :
+int lorenz_dvdt4gsl(double t, const double v[], double dvdt[], void *p)
+{
+  lorenz_dvdt((double *)p, t, (double *)v, dvdt);
+  return GSL_SUCCESS;
+}
        :
 struct ode lorenz_ode = {
-  3, 3, lorenz_init, lorenz_options, lorenz_dvdt
+  3, 3, lorenz_init, lorenz_options, lorenz_dvdt, lorenz_dvdt4gsl
 }, *ode = &lorenz_ode;
        :
```

（例題１）の解答例7-2/2

```
 void view_draw(struct view *v, void *o)
 {
   struct odeset *odes = (struct odeset *)o;
+  const gsl_odeiv_step_type *T = gsl_odeiv_step_rk4;
+  gsl_odeiv_step *s = gsl_odeiv_step_alloc(T, odes->ode->N);
+  gsl_odeiv_system sys =
+    { odes->ode->dvdt4gsl, NULL, odes->ode->N, (void *)odes->p };
+  double verr[odes->ode->N];
   int i, j;
   int x0, y0, x1, y1, c, ncolors = sizeof(v->colors)/sizeof(XColor);
   double velocity01, mat[3][3], qua[4];
@@ -374,11 +387,13 @@
     mat3_to_ang(mat, ang);
     ang_to_qua(ang, qua);
   }
+  for (j=0; j<odes->ode->N; j++)
+    odes->v1[j] = odes->v0[j];
   for (i=0; i<10000; i++) {
     double p[3], r[4];

-    ode_rk4(odes->p, odes->delta_t, odes->ode->N, odes->ode->dvdt,
-            odes->delta_t*i, odes->v0, odes->v1);
+    gsl_odeiv_step_apply(s, odes->delta_t*i, odes->delta_t, odes->v1, verr,
+                         NULL, NULL, &sys);
        :
+  gsl_odeiv_step_free(s);
 }
```

glx(OpenGL + X11)の利用

「glx(OpenGL + X11) の機能」
アニメーション、点・線・ポリゴン、視界、カラー、照光、混合・アンチエイリアス、ビットマップ、フォント、テクスチャマッピング、タイリングと曲面、セレクション等 OpenGL 及び X11 の全機能

『glx がシステムで使用可能かどうかの確認』

```
$ glxgears
$ glxinfo
$ xdpyinfo | less
```

```
         :
number of extensions: 26
         :
    BIG-REQUESTS
    DEC-XTRAP
    DOUBLE-BUFFER
    Extended-Visual-Information
    FontCache
    GLX
         :
```

（例題１）の解答例8-1/1

```
$ cd ../glx
$ less v_lorenz.c # glx を利用する
```

```
        :
#include <GL/glx.h>
#include <GL/gl.h>
        :
void view_draw(struct view *v, void *o)
        :
  glXMakeCurrent(v->display, v->window, v->ctx);
  glViewport(0, 0, v->w, v->h);
  glLoadIdentity(); glOrtho(-1/v->s, 1/v->s, -1/v->s, 1/v->s, -1/v->s, 1/v->s);
  glClearColor(1, 1, 1, 1); glClear(GL_COLOR_BUFFER_BIT);
  glRotated(v->deg[0],1,0,0); glRotated(v->deg[1],0,1,0); glRotated(v->deg[2],0,0,1);
  glBegin(GL_LINE_STRIP);
  for (i=0; i<10000; i++) {
        :
    x = (odes->v0[0] - odes->lb[0])/(odes->ub[0] - odes->lb[0])* 2 - 1;
    y = (odes->v0[1] - odes->lb[1])/(odes->ub[1] - odes->lb[1])* 2 - 1;
    z = (odes->v0[2] - odes->lb[2])/(odes->ub[2] - odes->lb[2])* 2 - 1;
    HSV2RGB((velocity01 - odes->lb[3])/(odes->ub[3] - odes->lb[3]), 1, 1, &r, &g, &b);
    glColor3d(r, g, b); glVertex3d(x, y, z);
    for (j=0; j<odes->ode->N; j++)
      odes->v0[j] = odes->v1[j];
  }
  glEnd();
  glFlush(); glXSwapBuffers(v->display, v->window);
        :
```

OpenGL + SDL の利用

「OpenGL + SDL の特徴」
ポータビリティが現時点では劣るものの、SDL がプラットフォームの違いを吸収するので、単純なコードで済む。☞ 類似の形態：OpenGL + GLUT

『SDL のインストール』

```
$ wget -N http://www.libsdl.org/release/SDL-1.2.11.tar.gz
$ tar tvzf SDL-1.2.11.tar.gz | less
$ tar xvzf SDL-1.2.11.tar.gz
$ cd SDL-1.2.11
$ ./configure --help | less
$ ./configure --prefix=$HOME/local
$ make
$ make install
```

『SDL のアンインストール』

```
$ make -n uninstall | less
$ make uninstall
```

（例題１）の解答例9-1/1

```
$ cd ../sdl+gl
$ less v_lorenz.c # OpenGL + SDL を利用する
```

```
#include <SDL.h>
#include <SDL_opengl.h>
        :
void view_init(struct view *v)
{
  if (SDL_Init(SDL_INIT_VIDEO) < 0)
    exit(1);
  v->info = SDL_GetVideoInfo();
  v->screen = SDL_SetVideoMode(v->w, v->h, v->info->vfmt->BitsPerPixel,
                               SDL_OPENGL);
}
void view_draw(struct view *v, void *o)
{
        :
  SDL_GL_SwapBuffers();
}
void view_loop(struct view *v, void *o)
{
        :
}
void view_term(struct view *v)
{
  SDL_Quit();
}
```

数値計算と３Ｄグラフィックス：まとめ

数値計算結果の理解や表現を助ける可視化技術 — 自らが理解を深めるには可視化された数値計算結果を積極的に観ることが重要である：

- ▷ **３次元空間に数値計算結果を可視化するための技術**
- ▷ **加えて、時間軸を数値計算に活用するアニメーション技術**
- ▷ **さらに、色空間を数値計算に活用する色管理技術**

応用として、いくつかの数値計算ルーチンを（自前で）実装するだけでなく、既存の数値計算ライブラリの利用として、 GPL(General Public License) **の** gsl(GNU Scientific Library) **のインストールと利用例を紹介した。**