数理解析研究所講究録
第561巻 1985年 69-78

muSIMP 上に実現されたマウスによる結び目作図システム

大阪大学理学部 落合豊行 (Mitsuyuki Ochiai)

1. 概観. 結び目が自明なものであるかどうかを決定することは一般的には非常に難しい。有効なアルゴリズムとして知られているのは、本間-落合-高橋の定理(1) を利用して証明された本間-落合の定理(2)のみであり、より一般的なアルゴリズムの出現が望まれる。筆者はこの問題と3ー球面判定のアルゴリズムを並行して研究しているが、これらの問題を考える上でどうしても必要な作業として、結び目の作図とかヘーゴード図式を描き、修正して、また描く、すなわち消しゴムとエンピツによる作業が伴う。現在のように、コンピュータの発達した社会では、これらの作業の大部分はコンピュータ・システムに移行でき、実際山田君(5)や児玉君(3)、筆者(4) によるシステムがあります。これらのシステムは、最近日本の社会に普及し始めたマウスを利用することにより、より利用価値のある使いがっての良いシステムにすることが可能です。

2. 結び目の作図システム。

このシステムは、二つの部分から構成されています； その第一はマイコン数式処理システムmuSIMP-83のソフトウェアー割り込みとして定義されたマシン・コード・ルーチン（この部分はPC-9801/E/FのグラフィクLIOを利用するコードと、マウス・ドライバーを利用するコード）、他の部分はmuMATHによるプログラムのコード。

システムをmuSIMPの上に実現した（現在実現されている程度のシステムのためにはBASIC で十分である）理由は、この数式処理システムの将来性にとても魅力をかんじていることと,muSIMPによるプログラミィングがBASICによるものと比べて容易であり,システムとしての拡張性が格段に優れているからです。別の理由は、この数式処理システムは位相的不変量を実用的レベルで計算可能であり、山田君や大阪大学の助手の村上君による、アレキサンダー多項式, ジョーンズ多

項式を計算するプログラムが既にあることにもよる。

プログラムは凡て巻末に掲載してあります、それらを参考にしてこのシステムの使用手順を述べます;このシステムは以下の機能から構成されています。

(1) 関数 KNOT(): 左のボタンを押しながら、マウスを動かすとマウスの動作の通りにある幅で結び目をかく、左のボタンを離すと書くことなしに移動 し、右のボタンを押すと、修正するスッテプにはいる、左ボタン押しながら修正し、左ボタンを離して右ボタン押すことによってこのスッテプを抜ける、次に色の問い合わせに答え、左ボタンを押しつずけて作業を継続するか、又は右のボタンを押して作業を終わる。

(2) 関数 BRIDGEKNOT(): この関数は４－橋結び目を、四本のアンダーパスを３から始まるパレット番号の色でそれぞれ，Ｘ軸又はＹ軸に平行に、マウスの左ボタンを押すごとに書く，Ｘ座標とＹ座標の変化の大きい方の動きに従う。四本のオーバパスは凡てパレット番号２で書かれる。結び目を書き終わると関数MODIFY()を呼び修正モードになる。結び目の線はグラフィク関数LINEのボックス塗り潰しを利用し、修正モードでの作業を容易にするため、オーバパスは凡て書いた後に黒い線の四角で囲っている。

(3) 関数 MODIFY(): この関数は結び目を変形、修正するもので、関数KNOT()を利用する部分と(2)の最後で述べた機能の部分にわかれる。

(4) その他諸ものの関数. （参照 巻末プログラム）

プログラムを見てもらうと、 理解されるようにプログラムの可読性と拡張性はmuSIMPで書いたことによりとても良い。

3. 自明な結び目について。

上に述べたシステムを利用して、簡単な結び目射影を持つ自明な結び目から始めて,より意味のある複雑な射影を持つ自明な結び目が簡単に構成出来るので

あるが、はたして次のような自明な結び目は存在するのか？
”交点数が任意に大きなウェーブを持たない４－橋結び目”
最近の結び目理論の話題の中心は、ジョーンズ多項式とそれに関する研究となるが,九大の金信君は次のことを示した;” 同じジョーンズ多項式を持つ異なる結び目が無限個存在する ”

それでは、ジョーンズ多項式が１である自明でない結び目は存在するのか？このような自明でない結び目を構成するのに、自明な結び目を利用することが可能であるように思われる。まず、あまり簡単すぎない自明な結び目射影L+を選びL+の一つの交点PでL-，L_0を考える、この時L_0 が自明な２－コンポーネント絡み目でL-が自明でない結び目であれば，L- は上に述べた条件を満たす結び目となる。このような方法で（あまり虫のよすぎる論法であるが）、構成出来るとすれば始めに述べた作図システムは強力な道具となる。幾つかの結び目（例えば森川君の自明な結び目の例）、に実際にこの作業をしてみればいかに強力がすぐに理解されると思います。

参考文献

(1) HOMMA-OCHIAI-TAKAHASHI, "An algorithm for recognizing S^3 in 3-manifolds with Heegaard splittings of genus two", Osaka J. Math.,17(1980)

(2) HOMMA-OCHIAI, "On Relations of Heegaard Diagrams and Knots", Math. Sem. Notes of Kobe Univ.,6(1978).

(3) 児玉宏児 ”ノットセオリーとコンピュータ ”, 数理科学 10 (1984)

(4) 落合豊行 ”コンピュータによる３－次元多様体の構成とグラフィクスによる処理” 数理科学 10 (1984)

(5) 山田修司 ” コンピュータによる結び目の作図 ”, 数理科学 10 (1084)

muSIMP によるプログラミィングは,

(6) D. Stoutemyer, muMATH SYMBOLIC MATHEMATICS PACKAGE USER MANUAL

グラフィクスに関する部分は,

(7) 井上智博, グラフィクス 解析マニュアル、秀和システムトレーディング

```
% Knot Drawing Function %

FUNCTION KNOT(COLOR), % Utility to draw a knot using curves %
  LOOP
    MSG("Left botton: cont. Right botton: exit"),
    WHEN MOUSE(), TRUE EXIT,
    MSG("Keyin color ? (Curve stage)"),
    COLOR : SCAN(),
    MOVE(1,COLOR),
    MSG("Please modify the knot :"),
    PRINTLINE("ERACE CP:"),
    ERACE(),
    PRINTLINE("ERACE CB:"),
    MOVE(1,0),
  ENDLOOP,
ENDFUN$

FUNCTION MSG(STRNG),
  PRINT(ASCII(27)),
  PRINT('"*"),
  PRINTLINE(STRNG),
ENDFUN$

% Line Drawing Function Using Mouse %

FUNCTION BRIDGEKNOT(COLOR,COUNT), % Utility to draw a 4-bridge knot %
  PRINT(ASCII(27)),
  PRINT('"*"),
  PRINTLINE("Left bottan sets current positin (CP)"),
  PRINTLINE("Right bottan sets last position (LP)"),
  PRINTLINE("Please set the start point: (Push right bottan)"),
  PRINTLINE("Go on the first stage:"),
  COLOR : 1,
  BDRAW(COLOR), % Draw knot projection %
  MSG("Go on the second stage:"),
  COLOR :3,      % Set first color %
  PRINTLINE("Underpath:"),
  COUNT : COLOR,
  LOOP
    WHEN COUNT = 7,  EXIT,
    PRINT("S:"),
    MOUSE(), % Push right botton, then LP=(REGISTER(2),REGISTER(3))
               is setted %
    PRINT(COUNT-2),
    LOOP
      WHEN MOUSE(), EXIT,
      % Push right botton, go out LOOP %
      % Push left  botton, then CP=(REGISTER(0),REGISTER(1)) is setted %
      FATLINE(REGISTER(2),REGISTER(3),REGISTER(0),REGISTER(1),COUNT,0),
    ENDLOOP,
    COUNT : COUNT + 1,
  ENDLOOP,
  MSG("Overpath:"),
  COLOR : 2, % Set color of overpath %
  COUNT : 1,
  LOOP
    WHEN COUNT = 5, EXIT,
```

```
    PRINT("S:"),
    MOUSE(),
    PRINT(COUNT),
    MOUSE(),
    FATLINE(REGISTER(2),REGISTER(3),REGISTER(0),REGISTER(1),COLOR,1),
    COUNT : COUNT + 1,
  ENDLOOP,
  MODIFY(),
ENDFUN$

FUNCTION MODIFY(COLOR), % Utility to modify 4-bidge knot %
  LOOP
    MSG("ERACE P"),
    ERACE(),
    KNOT(),
    MSG("Keyin color ? (Alphabet End)"),
    COLOR: SCAN(),
    WHEN NOT INTEGER(COLOR), TRUE EXIT,
    MSG("Go on the modifying stage:"),
    PRINT("S:"),
    MOUSE(),
    PRINT(COLOR),
    LOOP
      WHEN MOUSE(), EXIT,
      FATLINE(REGISTER(2),REGISTER(3),REGISTER(0),REGISTER(1),COLOR,1),
    ENDLOOP,
  ENDLOOP,
ENDFUN$

FUNCTION FATLINE(X1,Y1,X2,Y2,CLR,BOXFLAG,PX,PY),
  PXY(X1,Y1,X2,Y2),
  BLOCK
    WHEN ZERO(PY), REGISTER(2,X2), REGISTER(3,Y1+3),
      WHEN X1>X2, CHANGEX() EXIT EXIT,
    REGISTER(2,X1+3), REGISTER(3,Y2),
    WHEN Y1>Y2, CHANGEY() EXIT
  ENDBLOCK,
  WHEN ZERO(PY), LINE(X1,Y1,X2,Y1+3,CLR,2,1,CLR),
    WHEN BOXFLAG=1, LINE(X1-1,Y1-1,X2+1,Y1+4,0,1,0) EXIT EXIT,
  LINE(X1,Y1,X1+3,Y2,CLR,2,1,CLR),
  WHEN BOXFLAG=1, LINE(X1-1,Y1-1,X1+4,Y2+1,0,1,0) EXIT,
ENDFUN$

FUNCTION PXY(X1,Y1,X2,Y2),
  PX: ABS(X1-X2),
  PY: ABS(Y1-Y2),
  WHEN PX>PY, PY: 0  EXIT,
  PX: 0,
ENDFUN$

FUNCTION CHANGEX(TEMP),
  TEMP: X1,
  X1: X2,
  X2: TEMP,
ENDFUN$

FUNCTION CHANGEY(TEMP),
```

```
  TEMP: Y1,
  Y1: Y2,
  Y2: TEMP,
ENDFUN$

FUNCTION ERACE(),
  LOOP
    WHEN MOUSE(), EXIT,
    PAINT1(REGISTER(0),REGISTER(1),0,0),
  ENDLOOP,
ENDFUN$

FUNCTION BDRAW(COLOR),
  MOUSE(),
  LOOP
    WHEN MOUSE(), TRUE EXIT,
    SLINE(REGISTER(2),REGISTER(3),REGISTER(0),REGISTER(1),COLOR),
  ENDLOOP,
ENDFUN$

FUNCTION SLINE(X1,Y1,X2,Y2,CLR,PX,PY),
  PXY(X1,Y1,X2,Y2),
  WHEN ZERO(PY), LINE(X1,Y1,X2,Y1,CLR),
                 REGISTER(2,X2), REGISTER(3,Y1) EXIT,
  LINE(X1,Y1,X1,Y2,CLR),
  REGISTER(2,X1),
  REGISTER(3,Y2),
ENDFUN$

FUNCTION TRACEPOINT(COLOR),
  MOUSE(),
  LOOP
    WHEN MOUSE(), TRUE EXIT,
    LINE(REGISTER(2),REGISTER(3),REGISTER(0),REGISTER(1),COLOR),
    REGISTER(2,REGISTER(0)),
    REGISTER(3,REGISTER(1)),
  ENDLOOP,
ENDFUN$

RDS() $

A>
```

［グラフィクス関数の説明］

```
INIT:       グラフィクスLIOの初期化
GLOAD:      ディスクから画面データを読み込む
GSAVE:      画面をディスクに書き込む
HCOPY:      GP-700Mにカラーハードコピー
SCREEN:     グラフィクス画面のモード設定
VIEW:       描画領域の指定
COLOR1:     エリアの色の指定
COLOR2:     パレットの変更
CLS:        画面の消去
PSET:       ドットのセット，リセット
LINE:       直線，ボックスを書く
CIRCLE:     円，楕円を書く
PAINT1:     塗りつぶし
PAINT2:     タイルパターンによる塗りつぶし
GGET:       画面上のグラフィクスデータをメモリに読み込む
GPUT1:      グラフィクスデータの書き込み
GPUT2:      画面に漢字を表示
ROLL:       画面のスクロール
POINT2:     指定ドットの色を得る
```

［マウス関数の説明］

MOUSE()：　左のボタンでCP=(REGISTER(0),REGISTER(1))をセットして値"FALSE"で返る、右のボタンでLP=(REGISTER(3),REGISTER(4))をセットして値"TRUE"で返る。

MOVE(0,COLOR)：　左のボタンを押すごとに,COLORで点を打つ、右のボタンでもどる。

MOVE(1,COLOR)：　左のボタンを押すごとに、縦３、横３のボックスで塗りつぶす、右のボタンでもどる。

MOVE(2,COLOR)：　右のボタンを押た点と、左のボタンを押た点の間にボックスを書く。

MOVE(3,COLOR)：　右のボタンを押た点を中心に、左のボタンを押た点と原点との間のＸ座標，Ｙ座標で楕円を書く。

```
% muSIMP によるグラフィクス関数の定義 %
RADIX(16)$
PUTD('"GRAPH#",200)$      ; グラフィクス関数登録
PUTD('MOUSE,203)$         ; マウス関数１登録
PUTD('MOVE,206)$          ; マウス関数２登録
CSEG : REGISTER(0)$       ; コード・セグメント セーブ
RADIX(0A)$

FUNCTION CHECKREG(),                  |     FUNCTION INIT(),
  WHEN ZERO(REGISTER(4)),0 EXIT,      |       REGISTER(6,0),
  FALSE,                              |       "GRAPH#"(),
ENDFUN$                               |       CHECKREG(),
                                      |     ENDFUN$

FUNCTION GLOAD(),                     |     FUNCTION GSAVE(),
  REGISTER(6,0),                      |       REGISTER(6,0),
  "GRAPH#"(1),                        |       "GRAPH#"(2),
```

```
  CHECKREG(),                          |          CHECKREG(),
ENDFUN$                                |        ENDFUN$


FUNCTION HCOPY(),
  REGISTER(6,0),
  "GRAPH#"(3),
  CHECKREG(),
ENDFUN$

FUNCTION SCREEN(VAR1,VAR2,VAR3,VAR4),
% VAR1: 画面モード VAR2: 画面スイッチ VAR3: アクティブ画面
  VAR4: ディスプレイ画面 %
  REGISTER(6,1),
  "GRAPH#"(VAR1,VAR2,VAR3,VAR4),
  CHECKREG(),
ENDFUN$

FUNCTION VIEW(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6),
% VAR1: 左上のX座標 VAR2: 左上のY座標 VAR3: 右下のX座標
  VAR4: 右下のY座標 VAR5: 領域色       VAR6: 境界色 %
  REGISTER(6,2),
  "GRAPH#"(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6),
  CHECKREG(),
ENDFUN$

FUNCTION COLOR1(VAR1,VAR2,VAR3),
% VAR1: バックグラウンドカラー VAR2: ボーダーカラー
  VAR3: フォグラウンドカラー %
  REGISTER(6,3),
  "GRAPH#"(VAR1,VAR2,VAR3),
  CHECKREG(),
ENDFUN$

FUNCTION COLOR2(VAR1,VAR2),                    | FUNCTION CLS(),
% VAR1: パレット番号 VAR2: カラーコード %      |   REGISTER(6,5),
  REGISTER(6,4),                               |   "GRAPH#"(),
  "GRAPH#"(VAR1,VAR2),                         |   CHECKREG(),
  CHECKREG(),                                  | ENDFUN$
ENDFUN$

FUNCTION PSET(VAR1,VAR2,VAR3),
% VAR1: X座標 VAR2: Y座標 VAR3: パレット番号 %
  REGISTER(6,6),
  "GRAPH#"(VAR1,VAR2,VAR3),
  CHECKREG(),
ENDFUN$

FUNCTION LINE(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6,VAR7,VAR8),
% VAR1: 始点のX座標 VAR2: 始点のY座標 VAR3: 終点のX座標
  VAR4: 終点のY座標
  VAR5: パレット番号1 VAR6: 描画コード 0: 直線 1: ボックス
  2: ボックス 全体
  IF VAR6=0 OR 1 THEN VAR7: ラインスタイルスイッチ 0: 指定有り
  1: 指定なし          VAR8: ラインスタイル
  IF VAR6=2      THEN VAR7: パレット、タイルスイッチ 0: 指定なし
                            1: パレット指定有り 2: タイル指定有り
```

```
                        VAR8: パレット番号２（塗り潰す色）　%
  REGISTER(6,7),
  "GRAPH#"(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6,VAR7,VAR8),
  CHECKREG(),
ENDFUN$

FUNCTION CIRCLE(VAR1,VAR2,VAR3,VAR4,VAR5,
                VAR6,VAR7,VAR8,VAR9,VAR10,VAR11),
% VAR1: 中心点のＸ座標 VAR2: 中心点のＹ座標 VAR3: Ｘ軸方向の半径
  VAR4: Ｙ軸方向の半径 VAR5: パレット番号１（線色）
  VAR6: 描画コード ビット０: 開始点指定 ビット１: 開始点半径
  ビット２: 終了点指定 ビット３：終了点半径 ビット４: 開始点、
  終了点一致　ビット５: 塗り潰し指定 ビット６: タイル指定
  (０-> 有り, １-> なし)
  VAR7: 開始点のＸ座標 VAR8: 開始点のＹ座標 VAR9: 終了点のＸ座標
  VAR10: 終了点のＹ座標
  IF ビット５＝１ AND ビット６＝０ THEN VAR11: パレット番号２（領域色）
  IF ビット５＝１ AND ビット６＝１ THEN VAR11: タイルストリング文字数 %
  REGISTER(6,8),
  "GRAPH#"(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6,VAR7,VAR8,VAR9,VAR10,VAR11),
  CHECKREG(),
ENDFUN$

FUNCTION PAINT1(VAR1,VAR2,VAR3,VAR4),
% VAR1: 塗り潰し開始点のＸ座標 VAR2: 塗り潰し開始点のＹ座標
  VAR3: パレット番号（領域色） VAR4: パレット番号（境界色）%
  REGISTER(6,9),
  "GRAPH#"(VAR1,VAR2,VAR3,VAR4),
  CHECKREG(),
ENDFUN$

FUNCTION PAINT2(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6),
% VAR1: 塗り潰し開始点のＸ座標 VAR2: 塗り潰し開始点のＹ座標
  VAR3: パレット番号（境界色） VAR4,5,6: タイルパターン %
  REGISTER(6,10),
  "GRAPH#"(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6),
  CHECKREG(),
ENDFUN$

FUNCTION GGET(VAR1,VAR2,VAR3,VAR4,VAR5),
% VAR1: 読み込む領域の左上のＸ座標 VAR2: 読み込む領域の左上のＹ座標
  VAR3: 読み込む領域の右下のＸ座標 VAR4: 読み込む領域の右下のＹ座標
  VAR5: 格納領域の長さ＝4+((VAR3-VAR1+8)/8*3)*(VAR4-VAR2+1) %
  REGISTER(6,11),
  "GRAPH#"(VAR1,VAR2,VAR3,VAR4,VAR5),
  CHECKREG(),
ENDFUN$

FUNCTION GPUT1(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6,VAR7),
% VAR1: 書き込む領域の左上のＸ座標 VAR2: 書き込む領域の左上のＹ座標
  VAR3: 格納領域の長さ VAR4: 描画モード ０: ＰＳＥＴ １: ＰＲＥＳＥＴ
  ２: ＯＲ ３: ＡＮＤ ４: ＸＯＲ
  VAR5: カラースイッチ ０: 指定なし １: 白と黒に色をつける
  VAR6: 白につける色 VAR7: 黒につける色 %
  REGISTER(6,12),
```

```
  "GRAPH#"(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6,VAR7),
  CHECKREG(),
ENDFUN$

FUNCTION GPUT2(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6,VAR7),
% VAR1: 表示領域の左上のＸ座標 VAR2: 表示領域の左上のＹ座標
  VAR3: ２バイト文字コード
  VAR4: 描画コード ０: ＰＳＥＴ １: ＰＲＥＳＥＴ ２: ＯＲ ３: ＡＮＤ
  ４: ＸＯＲ VAR5: カラースイッチ VAR6: 字につける色 VAR7: 背景色 %
  REGISTER(6,13),
  "GRAPH#"(VAR1,VAR2,VAR3,VAR4,VAR5,VAR6,VAR7),
  CHECKREG(),
ENDFUN$

FUNCTION ROLL(VAR1,VAR2,VAR3),
% VAR1: 上下方向のドット数 正: 上にスクロール 負: 下にスクロール
  VAR2: 左右方向のドット数 正: 左にスクロール 負: 右にスクロール
  VAR3: クリアフラッグ ０: パレット番号０でクリア
  １: バックグラウンドカ ラーでクリア %
  REGISTER(6,14),
  "GRAPH#"(VAR1,VAR2,VAR3),
  CHECKREG(),
ENDFUN$

FUNCTION POINT2(VAR1,VAR2),
% VAR1: 色を得るドットのＸ座標 VAR2: 色を得るドットのＹ座標 %
  REGISTER(6,15),
  "GRAPH#"(VAR1,VAR2),
  CHECKREG(),
ENDFUN$

RDS()$
```

muSIMPのソフトウェー割り込みとして定義されたマシン・コード関数のリストは、この講究録に掲載するには量があまりに多く、かつその必要もないと思うのでやめます。その内容に興味のある読者は直接大阪大学の落合まで連絡してください。

muSIMP-83/muMATH-83 は The Software House の登録商標です。