

0. 序 章

0.1 表 記 法

0.1.1 入力と出力の表し方

本書では *MATHEMATICA*[®] のノートブックへの入力と、その出力をコンピュータ上における表現と同じように示す。入力はプロンプト $In[n]:=$ で始まるクーリエ書体の太字で表し、その出力は $Out[n]=$ で始まる。ここに、番号 n は章毎の通し番号とした。オンラインヘルプの $?$ で始まる入力に対する出力セルでは $Out[n]=$ が現れない。本書に書かれた入力コマンドを入力し (入力セルをアクティブにした後に `SHIFT` + `RET` により) 実行することをすすめる。

0.1.2 諸 記 号

説明の途上にある種々の記号は次のような意味を持たせてある。

- 💡 : ヒント
- ⚙️ : *MATHEMATICA*[®] の文法の簡単な説明
- 📖 : ノートブックで定義した変数などの説明
- ⚠️ : 注意
- 📖 : *MATHEMATICA*[®] による演習の指示
- 📖 : *MATHEMATICA*[®] によらない学習の指示

0.2 コンピュータの環境調査

MATHEMATICA[®] の性能をフルに活かすには、インターネットを利用して他のユーザが作成したノートブック (拡張子 nb) やパッケージ (拡張子 m) を利用するのがよい。そのためには *MATHEMATICA*[®] がインストールされた自分のコンピュータの構成を知っておかねばならない。

本書は MS-Windows 上で *MATHEMATICA*[®] を利用する筆者の環境を例に説明して行く。*MATHEMATICA*[®] を起動し、カレントディレクトリーと、パスの通っているディレクトリーを調べておく。こうすることによって、*MATHEMATICA*[®] のしくみと使い方の理解がすこしづつ深まっていくであろう。

0.2.1 ディレクトリー

まず、「ディレクトリー (Directory)」に関して用意されている関数と大域変数を知ろう。

```
In [1] := ? *Directory*
```

```
System'
```

```
CopyDirectory RenameDirectory
```

```
CreateDirectory ResetDirectory
```

```
DeleteDirectory SetDirectory
```

```
Directory $HomeDirectory
```

```
DirectoryName $InitialDirectory
```

```
DirectoryStack $LaunchDirectory
```

```
HomeDirectory $PreferencesDirectory
```

```
NotebookDirectory $RootDirectory
```

```
ParentDirectory $TopDirectory
```

- 💡 $In[1]$ で用いたように、知りたい記号の前に“?”を書くと必要な情報が得られる。
- 💡 メニュー：ヘルプをクリックしてヘルプブラウザを開き、組み込み関数などに関する詳細な情報が得られる。*MATHEMATICA*® ブックの膨大な内容も検索することができる。
- ⚙️ “*”はワイルドカード（キャラクタ）であり、任意の文字列を意味する
- ⚙️ “'”はコンテキストの文字列の終わりを表す。*MATHEMATICA*® ではコンテキストはパッケージに対応させている。たとえば、?`System`CopyDirectory` と `CopyDirectory` とは、`System`` が使える状態では同じである。
- ⚙️ “\$”で始まる変数は大域変数であり、デフォルト値が割り当てられている。
- 🔍 この方法で $In[1]$ の出力のいくつかを調べてみよ。
- 🔍 $In[1]$ の働きを知るために、このセルをクリックしてアクティブにし、メニュー：“セル / 形式変換 / 標準形”を選んでみよ。
- 🔍 “\$Packages”を入力し、現在使われているパッケージを調べてみよ。
- 🔍 知りたい記号の前に??と書くとうなるか。試みてみよ。
- 💡 バージョン 4.1 以降では、 $In[1]$ に対する出力の各項目が選択してクリックできるようになっていて、?によるものと同じ情報が得られる。また、?の機能が拡充されている。項目がそれぞれのヘルプブラウザにハイパーリンクされ、出力の右下に現れる 詳細をクリックすると、関連のヘルプブラウザが開く。

各自の環境は*MATHEMATICA*® のインストールのしかたによって異なる。以下は *MATHEMATICA*® コマンドにより筆者のコンピュータの環境を調査した結果であ

る。各自が同様の調査をして、今後の作業でディレクトリー名の読み替えができるように準備しておく。

```
In[2] := Directory[]
```

```
Out[2] = C:\Program Files\Wolfram
         Research\Mathematica\4.1
```

```
In[3] := ?Directory
```

Directory[] は、現行のディレクトリを返す。 詳細

✳ 関数は大文字で始まり、大括弧 [] で引数をくくって用いる。

△ Unix ではディレクトリーの区切りを表す \ が / となっている。

0.2.2 ファイルとパス

同様に「ファイル (File)」に関して調べてみよう。

```
In[4] := ? *File*
```

```
System'
```

```
ContextToFilename FileName
```

```
ContextToFileName FileNameDialogSettings
```

```
CopyFile FileNames
```

```
DeleteFile FileType
```

```
EndOfFile GetFileName
```

```
File IncludeFileExtension
```

```
FileBrowse RenameFile
```

```
FileByteCount SetFileDate
```

```
FileDate SetFileLoadingContext
```

```
FileFormat ToFileName
FileInformation $PasswordFile
```

大域変数\$Path はファイルの管理をする上で把握しておくべき重要な大域変数である。大域変数\$Path に含まれるディレクトリーはつぎの例のように増やすことができる。

```
In[5] := ?$Path
```

\$Path は、外部ファイルを検索するディレクトリのデフォルトリストを与える。詳細

```
In[6] := AppendTo[$Path,
  "C : \Program Files\Wolfram Research\
  Mathematica\4.1\inagaki"];
```

0.3 対話的な利用

MATHEMATICA® には多くの関数が用意されているので、C や Fortran では数十行以上の入力を要したものが、一行だけの入力で出力が得られることが多い。数行以下の入力毎に実行させ、期待通りの結果が得られるまで入力を改良して、前に進むことができる。新しくプログラムを書くときは、このように1コマンドごとに実行し、対話的に使用するのがよい。用意されている関数の数は次のようにして知ることができる。

```
In[7] := Names["System`*"];
  Length[%]
Out[7] = 1861
```

- ✪ 行末にセミコロン; をつけると結果の表示を抑圧できる。また、複数の式をセミコロンでつないで1行に書くことができる(複合式)。

便利さには常に不便さもつきまとう。対話的利用で、以前に実行した結果は残っているの、困ることもある。このような場合は次のコマンドを先頭に書いておくとよい。この意味はオンラインヘルプで調べておこう

```
In[8] := Clear[Global`*]
```

0.4 パレットの作成

MATHEMATICA® の入力を援助する機能として、パレットがある。バージョン 2.x はコンソール入力しかできなかったが、バージョン 3.x 以上ではパレットがいくつか用意されていて、StandardForm(標準形)の入力がマウスとキーボードを使ってできる。また、ユーザが自分の使用目的にあったパレットをつくることもできる。筆者がつくったパレットの例を図 0.1 と図 0.2 に示す。FieldLines.nb は 0.8.1 に説明するベクトルの力線を描くためのコマンド用のパレットである。

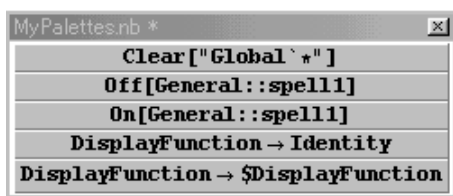


図 0.1: MyPalettes.nb

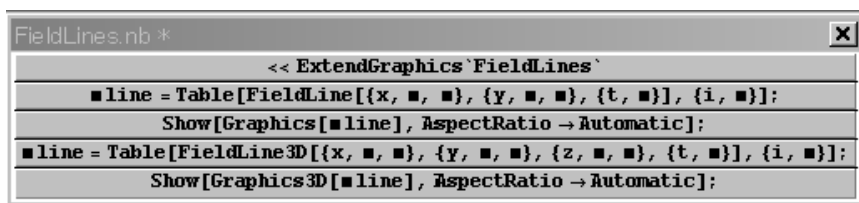


図 0.2: FieldLines.nb

各自の好みにあったパレットをつくって使用しよう。つくり方は以下の通りである。

1. メニュー：“入力/表・行列・パレットの作成”をクリックして現れる入力ウィンドーでパレットを選び、行・列数を記入する。筆者の上記の例ではそれぞれ5、1とした。
2. 新しいセルに現れた空のパレットに各自の好みの入力を書き入れる。このとき、入力待ちのスペースは *MATHEMATICA* の名前付き文字で、\[SelectionPlaceholder] により書き入れることができる。
3. 上のセルをアクティブにし、メニュー：“ファイル/パレットの作成”をクリックすると、パレットが別のウィンドーとして現れて使えるようになる。
4. このパレットを保存し、次回の *MATHEMATICA* の使用時にも使えるようにするには、

```
C:/Program Files/Wolfram Research/Mathematica
/4.1/SystemFiles/FrontEnd/Palettes/
```

ディレクトリーに、たとえば MyPalettes.nb の名前で保存しておく。次回には、メニュー：“ファイル/パレット”を開くと、MyPalettes の名が現れるので、これをクリックすると使える。なお、前回の終了時に開かれていたパレットは、*MATHEMATICA* 開始時に（バージョン 4.0 以降では同じ位置に）現れて便利である。

0.5 グラフィックスの基礎知識

本書は *MATHEMATICA* のグラフィックス機能を活用するので、その基本用語をまとめておく。

点、線、面などの図形の基本要素を表す関数をグラフィックス・プリミティブという。グラフィックス・プリミティブには

```
Point, Line, Polygon, Rectangle, Circle, Disk,
Cuboid, Text, Raster
```

がある。更に、パッケージ `Graphics`Shapes`` をロードするとつぎのものが使える。

`Cone` , `Cylinder` , `DoubleHelix` , `Helix` , `MoebiusStrip` ,
`Sphere` , `Torus`

このパッケージには、図形基本要素を変換するつぎの関数も用意されている。

`AffineShape` , `RotateShape` , `TranslateShapem` , `WireFrame`

これらをディスプレイに表示するにはグラフィックス・プリミティブを関数

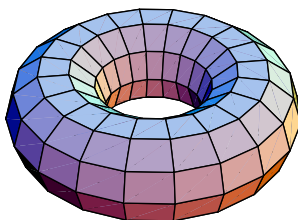
`Graphics` , `Graphics3D` , `GraphicsArray`

のいずれかの関数によりグラフィックス・オブジェクトに変換し、関数

`Show`

を用いる。このとき、関数のオプションを用いることができる。以下の例では3次元グラフィックスでトーラスを描いている。オプションの位置、種類を知っておこう。

```
In [9] := << Graphics`Shapes`
         Show[Graphics3D[Torus[1, 0.5, 20, 10]], Boxed → False]
```



```
Out [9] = -Graphics3D-
```

☞ <<は関数 `Get` の省略形である。ファイルが既に呼び出されているときも、読み直す。これを避けるには関数 `Needs` を使い、`Needs["Graphics`Shapes`"]` のようにすれば良い。

数学関数をプロットするための、数学関数からグラフィックスオブジェクトを作成し、表示するまでの一連の手続きを自動的に行なう関数としてつぎのものがある。

```
Plot, ParametricPlot, ListPlot, ContourPlot,
DensityPlot, Plot3D, ListPlot3D,
ListContourPlot, ListDensityPlot
```

更に、パッケージ Graphics`Graphics`には両対数グラフにプロットするための関数、LogLogPlot、など多数の関数が用意されている。その他、グラフィックス関係のパッケージには

```
Graphics`Graphics3D`, Graphics`ImplicitPlot`,
Graphics`Legend`, Graphics`MultipleListPlot`,
Graphics`ParametricPlot3D`, Graphics`PlotField`,
Graphics`PlotField3D`, Graphics`Polyhedra`,
Graphics`Spline`, Graphics`SurfaceOfRevolution`,
Graphics`ThreeScript`
```

がある。

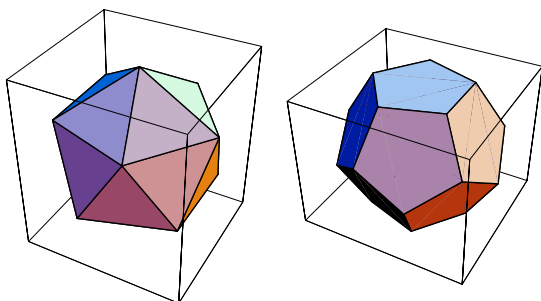
本書でこれら全てを用いるのではないが、興味があったらオンラインヘルプで意味と使用法を調べてみるとよい。

0.6 データファイルの作成

原点から 3 次元的に等しい角度間隔で出る 12 本及び 20 本のベクトルのリストを作成、保存する。

☞ *MATHEMATICA* 上で以下のコマンドを実行しよう。

```
In[10] := << Graphics`Polyhedra`
N[Vertices[Icosahedron]] >> "vt12.dat"
N[Vertices[Dodecahedron]] >> "vt20.dat"
Show[GraphicsArray[
{Graphics3D[Icosahedron[]], Graphics3D[Dodecahedron[]]}
]]
```



Out[10]= -GraphicsArray-

- ✪ *MATHEMATICA*[®] はリストを多用する。リスト $\{a, b\}$ に関数 f を作用させたとき、 $\{f(a), f(b)\}$ が出力されるとき、関数 f は `Listable` であるという。
- ✪ `Listable` でない関数のまま同様に使う方法として、関数 `Map` あるいはその省略形 `@` による方法がある。調べてみるとよい。

☞ ?? `Sin` を入力し、関数 `Sin` の属性を調べてみよ。

0.7 パッケージの作成

パッケージを作れるようになれば *MATHEMATICA*[®] のユーザとして一人前である。3次元の座標軸を描く簡単なパッケージを作ることによって、その一端に触れてみよう。 *MATHEMATICA*[®] のディレクトリーの中にパッケージを作るためのテンプレートがあるので、これをもとに作るのが良い。筆者の環境では、

```
C:/Program Files/Wolfram Research/Mathematica
/4.1/AddOns/ExtraPackages/ProgrammingInMathematica
/Template.nb
```

である。これを開き、その記述例に倣って書き込めばできあがる。以下は、その一例である。

```

BeginPackage["MyPackages`axes`"]
axes::usage = "axes.m is a package to create
              3D Cartesian coordinate axes by calling
              the function coord."
coord::usage = "coord[a,b,c] returns the 3D
              graphics primitive representing the Cartesian
              coordinate axes with the lengths a,b,c,
              in bpth x,y,z and -x,-y,-z directions"
Begin["Private`"]
  arrow3d[p1_, p2_, p3_, p4_] :=
    Line[{p1,p2,p3,p4,p2}]
  coord[a_,b_,c_] :=Module[
    {d,d2,d3,x1,y1,z1,xx,yy,zz,oo},
    d=0.025*Min[a,b,c];d3=3*d;
    x1=arrow3d[{-a,0,0},{a,0,0},{a-d3,d,0},
              {a-d3,-d,0}];
    y1=arrow3d[{0,-b,0},{0,b,0},{d,b-d3,0},
              {-d,b-d3,0}];
    z1=arrow3d[{0,0,-c},{0,0,c},{d,0,c-d3},
              {-d,0,c-d3}];
    xx=Text["x",{a+d3,0,0};
    yy=Text["y",{0,b+d3,0}];
    zz=Text["z",{0,0,c+d3}];
    oo=Text["O",{-d,-d,-d}];
    coord={oo,x1,xx,y1,yy,z1,zz}
  ]
End[]
EndPackage[ ]

```

💡 用法：“usage:…”を日本語で書く場合には適当なエディターを用いるのが無難である。

💡 *MATHEMATICA* で書く場合には、保存の方法に注意する必要がある。メニュー：“ファイル/特別な形式で保存/パッケージ形式”を選び、保存するディレクトリーとファイル名を記入する。念のため、保存されたファイルをエディターで読んでみよう。全体が(*と*)により囲まれてコメントアウトされていることがある。このときは、(*と*)を消して保存し直しておく。
 なお、本書ではこの方法でプログラム中にコメントを書くことがある。“書式/

スタイル/Input”のスタイルで書いているが、読者が同様にコメントを書く必要があるときは“書式/スタイル/Text”のスタイルのコメント用のセルをつくり、プログラムと分離した方が実行が早いと言われている。

- ☞ 保存するディレクトリとして、パスの通ったディレクトリー
C:/Program Files/Worfram Research/Mathematica/4.1
/AddOns/ExtraPackages
の中に、MyPackages ディレクトリーをつくっておき、ここに `axes.m` の名前で保存しておくことにする。
- ☞ Unix では、このディレクトリーは `/usr/local/mathematica/AddOns/ExtraPackages/` となる。多人数が共用する場合には、管理者に依頼して、一つだけ保存しておくことと良い。利用者個々にファイルを保存したいときは自分の適当なディレクトリーに保存し、パスを通しておく。

0.8 MathSource の利用

本書では目に見えない電磁界の理解を深めるために、電界や磁界の力線を多用する。この目的のためには *MATHEMATICA*[®] に組み込まれているパッケージのみでは満足できない。このようなとき、Wolfram Research Inc. の無料アーカイブサービス MathSource を検索することを薦める。MathSource は世界の *MATHEMATICA*[®] ユーザが作成したパッケージやノートブックが分野別に分類して、登録されている。本書では下記の二つを使うので、ダウンロードして使えるように準備しておこう。ダウンロードできるインターネットの URL は時々変更がある。現時点（平成 15 年 5 月）では

<http://library.wolfram.com/infocenter>

を開く。*MATHEMATICA*[®] に関する諸々の情報源への案内がある。Main Collection の中の MATHSOURCE: PACKAGES AND PROGRAMS をダブルクリックし、MathSource のページを開く。探索 (Search) のボックスが上部に現れる

ので、ここに欲しいパッケージ名を入力するとダウンロードできるページに案内される。

0.8.1 ExtendGraphics

これは、*MATHEMATICA*^{*} の開発者の一人である Tom Wickham-Jones 氏が作成したパッケージである。内容は、同氏の著書：

Mathematica Graphics: Techniques and Applications.
Tom Wickham-Jones, TELOS/Springer-Verlag 1994.

に詳述されているが、グラフィックスの機能を大幅に拡張するもので、我々は特に、その中の fieldIns.m を多用する。これは 2 次元、および 3 次元のベクトルの力線を描くためのパッケージである。これを使えるようにするために、

ExtendGraphics30.zip(156.7Kb)

をダウンロードし、解凍する。すると、ディレクトリー ExtendGraphics がつくられ、その中に 38 個のパッケージとノートブック、実行ファイルなどが現れる。その中の FieldLines.m を適当なエディターで開き、次の修正を行う。第 64, 90, 118, 144 行目の

```
t2 = Part[ sol, 1, 0, 1, 2]; を t2 = Part[ sol, 1, 0, 1, 1, 2];
```

に修正する。そして、38 個のパッケージファイルを一括して

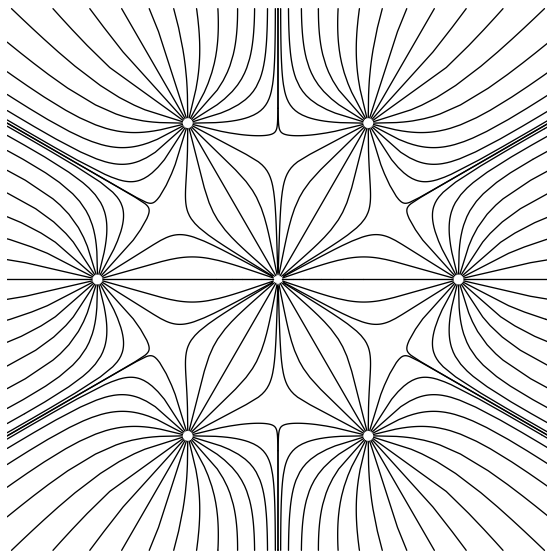
```
C:/Program Files/Wolfram Research/Mathematica/4.1  
/AddOns/ExtraPackages/ExtendGraphics
```

に保存しておこう^{*1}。この FieldLines.m の使用例を下に示す。この例は、正六角形の頂点に等しい大きさの点電荷があるときの電気力線を求めたもので

^{*1} Unix では/usr/local/mathematica/AddOns/ExtraPackages/ExtendGraphicsとなる。

ある。

```
In[11] := << ExtendGraphics`FieldLines`
xn[n_] := Cos[2 π (n - 1)/6];
yn[n_] := Sin[2 π (n - 1)/6];
r[x_, y_, n_] :=
  Sqrt[(x - xn[n])^2 + (y - yn[n])^2];
ex = Sum[(x - xn[n])/r[x, y, n]^3, {n, 1, 6}];
ey = Sum[(y - yn[n])/r[x, y, n]^3, {n, 1, 6}];
eline = Table[
  FieldLine[{x, ex, xn[n] + 0.03 Cos[i π/12]},
    {y, ey, yn[n] + 0.03 Sin[i π/12.]},
    {t, 1}], {i, 24}, {n, 6}];
Show[Graphics[
  eline, PlotRange → {{-1.5, 1.5}, {-1.5, 1.5}},
  AspectRatio → Automatic]
```



Out[11]= -Graphics-

※ $(x_n[n], y_n[n])$ は正六角形の n 番目の頂点の座標

⌘ `r[x,y,n]` は点 (x,y) と n 番目の頂点の間の距離

⌘ `ex,ey` は電界の x 成分と y 成分

⌘ ? `FieldLines` と入力すると

FieldLine::usage =

"FieldLine[x, ex, x0, y, ey, y0, t, t1] will calculate

the field line from the field ex, ey, starting at x0,y0,

of length t1. x and y are the variables of the field and

t is the variable of length down the trajectory."

と使用法の説明が帰ってくる。この場合、力線の始点 $(x0,y0)$ として、 $(xn[n], yn[n])$ から 0.03 の距離をおく 15 度の角度間隔の点としている。力線を描くアルゴリズムは媒介変数 t を用いる `ParametricPlot` によっている。 t の最大値 $t1$ は試行錯誤で選択する。ここでは 1 としている。

⌘ `eline` は力線に沿う点の座標のリストである。関数 `Show[Graphics[*]]` により表示させる。オプション `PlotRange` は描く範囲を指定するために用いている。

0.9 LiveGraphics3D

これはドイツの Martin Kraus 氏が作成した Java アプレットである。*MATHEMATICA*® により作成した 3 次元グラフィックスを Web 上に表示し、インタラクティブに縮小、拡大、回転、視点の移動、ステレオ表示を可能とする。この Java アプレットは非商用に公開されたフリーソフトウェアである。この class を用いることで、Java アプレットの知識を持たなくても引数を指定するだけでアプレットを実行することができ、操作が簡単である。

MS-Windows 版と Macintosh 版の *MATHEMATICA*® 4.x には `RealTime3D` パッケージが試験的に導入されていて、3 次元グラフィックスを回転させることができる*2。同様な機能を持ったソフトウェアに `VRML` や `JavaView` があるが、本書では筆者が経験上、最も使い勝手の良いと思う `LiveGraphics3D` を利用しよう。

*2 Unix 版にはバージョン 4.1 から導入された。

このために、Martin Kraus 氏によるホームページ：

<http://wwwvis.informatik.uni-stuttgart.de/kraus/LiveGraphics3D/index.html>

を開く。このページには前述の、MathSource から辿ることもできる。

download LiveGraphics3D 1.30 をクリックして **live.jar**(79Kb) をダウンロードする。

次に、read the documentation をクリックしてドキュメントを開き、Producing Graphics with Mathematica の節にリンクされた **LiveGraphics3D.m**(28Kb) をダウンロードする。また、このドキュメンテーション自体を保存（完全）しておくともマニュアルとして使用できるので便利である。

live.jar は Java1.1 applet であり、必須のファイルである。これをパスの通ったディレクトリーに置く。筆者は *MATHEMATICA*[®] の作業ディレクトリー：C:/Program Files/Wolfram Research/Mathematica/4.1 に置いている。

LiveGraphics3D.m は *MATHEMATICA*[®] により得る Graphics3D オブジェクトを操作し、変換する関数を含むパッケージである。これを **axes.m** と同じディレクトリーに保存しておく。

0.9.1 使用方法

0.9.1.1 Graphics3D オブジェクトの InputForm ファイルの作成

MATHEMATICA[®] 上でまず、"LiveGraphics3D.m"をロードする。すなわち

```
<<MyPackages`LiveGraphics3D`
```

を書いておく。次に、Graphic3D オブジェクトを作成する。例えば

```
g=Plot3D[Sin[x y],{x,-Pi,Pi},{y,-Pi,Pi}];
```

として、Graphic3D オブジェクト"g"を作成する。次に g から、次の命令により適切な InputForm からなるファイル"x.g3d"を作業中のディレクトリーに作成する。

```
WriteLiveForm["x.g3d",g];
```

0.9.1.2 HTML ページの作成

`documentation.html` の Quick Start の節に書かれている例を参考に HTML ファイルを作成する。例えば、下記のような `lg3d.html` を作成し、*MATHEMATICA*[®] の作業ディレクトリーに置く。

```
<HTML>
<APPLET ARCHIVE="live.jar" CODE="Live.class"
WIDTH=600 HEIGHT=600 ALIGN=ABSMIDDLE>
<PARAM NAME=BGCOLOR VALUE=#FFFFFF>
<PARAM NAME=MAGNIFICATION VALUE=1.>
<PARAM NAME=INPUT_FILE VALUE= "x.g3d">
</APPLET>
</HTML>
```

ファイル名“`x.g3d`”は上で作成した適当な `InputForm` のファイルである。なお、何回も上の方法でファイルをつくるとき、新しいファイルで上書きされる。永久保存したいときは、別に保存用のディレクトリーを用意し、“`x.g3d`”と“`lg3d.html`”を各グラフィックスに適切な名前をつけて保存しておくといよい。

0.9.1.3 Web browser で HTML ファイルを開き、3 次元画像を視る。

- 左マウスドラッグ：画像内の軸のまわりの回転
- 左マウスをドラッグしながら離す：画像内の軸のまわりのスピン（回転を続ける）
- SHIFT キー + 垂直ドラッグ：ズーム
- SHIFT キー + 水平ドラッグ：画像に垂直な軸の周りの回転
- CONTROL キー + 垂直ドラッグ：焦点距離の変化
- CONTROL キー + 水平ドラッグ：ステレオ効果の長さの変化
- 右マウス垂直ドラッグ：グラフィックスの部分分解
- “o”キー：Java コンソールへのパラメータ表示
- “s”キー：単一画像とステレオ画像のトグル
- HOME キー：元画像の復元（スピンしていない）

0.9.1.4 Java コンソールの開き方

- Netscape の場合
「Communicator」メニュー - の「ツール」を開き、その最下にある「Java Console」を選択する
- Internet Explorer の場合
「表示」メニュー - から「Java コンソール」を選択する

以下は LiveGraphics3D の使用例である。0.8.1 で用意したパッケージ：“FieldGraphics”を用いて、正負の二つの点電荷のつくる電気力線を 3 次元的に描いたものである。また、0.6 で用意したファイル `vt20.dat` と、0.7 で作ったパッケージ `axes.m` を利用している。Gaphics3D を用いると、電気力線の世界に入り込む仮想体験をすることができる。図 0.3 は次のノートブックにより作成した `x.g3d` を、`lg3d.html` を介してブラウザ上で見ているところである。

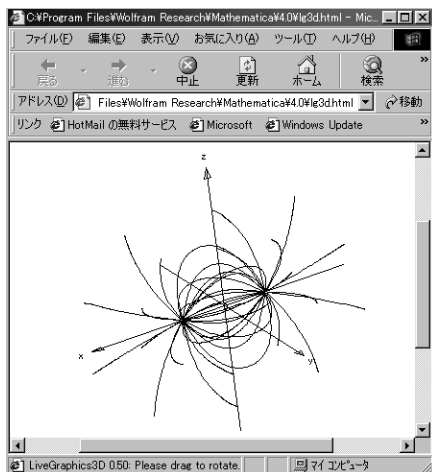
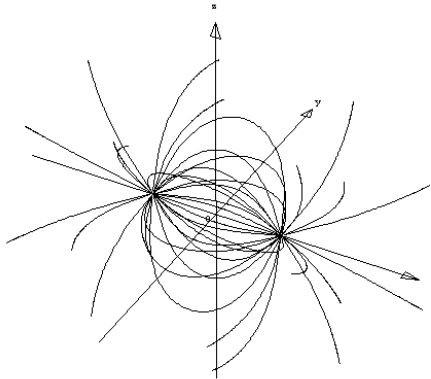


図 0.3: LiveGraphics3D の表示例

```

In[12] := << ExtendGraphics`FieldLines`
ex = (x + 1)/Sqrt[(x + 1)^2 + y^2 + z^2]^3 -
      (x - 1)/Sqrt[(x - 1)^2 + y^2 + z^2]^3;
ey = y/Sqrt[(x + 1)^2 + y^2 + z^2]^3 -
      y/Sqrt[(x - 1)^2 + y^2 + z^2]^3;
ez = z/Sqrt[(x + 1)^2 + y^2 + z^2]^3 -
      z/Sqrt[(x - 1)^2 + y^2 + z^2]^3;
dir = Get["vt20.dat"];
eline1 = Table[FieldLine3D[
  {x, -ex, 1 + 0.05 dir[[i, 1]]},
  {y, -ey, 0.05 dir[[i, 2]]},
  {z, -ez, 0.05 dir[[i, 3]]}, {t, 4}], {i, 1, 20}];
eline2 = Table[FieldLine3D[
  {x, ex, -1 + 0.05 dir[[i, 1]]},
  {y, ey, 0.05 dir[[i, 2]]},
  {z, ez, 0.05 dir[[i, 3]]}, {t, 4}], {i, 1, 20}];
<< MyPackages`axes`
xyzaxes = coord[3, 3, 3];
g3d = Show[Graphics3D[{xyzaxes, eline1, eline2}],
  AspectRatio -> Automatic, Boxed -> False]

```



```
Out[12]= -Graphics3D-
```

```

In[13] := << MyPackages`LiveGraphics3D`
WriteLiveForm["x.g3d", g3d];

```