4. 常微分方程式の解析

4.1 常微分方程式の厳密解

簡単なバネ-マス系の運動方程式

$$m\left(\frac{d^2}{dt^2}x(t)\right) = -k\,x(t) - 2\,m\,c\left(\frac{d}{dt}x(t)\right)$$

を Maple の機能を使って解きます。

この時、x の初期値を x0 、質点の初速度を V0 とします。

```
> restart:
```

微分方程式 eq と初期値 ini を定義します。

```
> eq:=m*diff(x(t),t,t)=-k*x(t)-2*m*c*diff(x(t),t);
> ini:=D(x)(0)=v0,x(0)=x0;
```

コマンド dsolve で微分方程式を数式的に解きます。

```
> ans:=dsolve([eq,ini],x(t));
```

解から右辺を取り出し Fortran のコードに変換します。

```
> pl:=rhs(ans):
> CodeGeneration[Fortran](pl);
```

パラメータの定義とプロット

```
> data1:={m=1.0,c=100,k=40,x0=0.1,v0=0.0};
> subs(data1,pl);
> plot(%,t=0..10);
```

複数のパラメータの設定とプロット

```
> data2:={m=1.0,c=6.3,k=40,x0=0.1,v0=0.0};
  data3:={m=1.0,c=0.5,k=40,x0=0.1,v0=0.0};
  data4:={m=1.0,c=0,k=40,x0=0.1,v0=0.0};
> plot([subs(data1,pl),subs(data2,pl),subs(data3,pl),subs(data4,
  pl)],t=0..10,color=[red,blue,green,navy]);
```

4.2 常微分方程式の数値解

dsolve で解をデータとして求める方法

dsolve コマンドで type=numeric オプションを指定すると、微分方程式の数値解を計算します。

ここでは、連立方程式

$$\frac{d}{dt}x(t) = y(t),\ \frac{d}{dt}y(t) = x(t) + y(t)$$

の初期値を指定し、解を求めます。

```
> restart;
> ff:=dsolve({diff(x(t),t)=y(t),diff(y(t),t)=x(t)+y(t),x(0)=2,y(0)
  =1},{x(t),y(t)},type=numeric);
```

結果は proc ... end proc のように関数として返されます。

引数を指定して実行すると実際の値を得ることができます。

```
> ff(1.0);
```

積分アルゴリズムに Runge-Kutta 法や Gear 法など、任意のものを指定することもできます。

設定を省略すると Runge-Kutta Fehhlberg 法を使用します。

(詳しくはオンラインヘルプ ?dsolve,numeric をご参照ください。)

積分手法を Runge-Kutta 4次、刻みを 0.1 に指定した例

```
> ff2:=dsolve({diff(x(t),t)=y(t),diff(y(t),t)=x(t)+y(t),x(0)=2,y
```

```
    (0)=1},{x(t),y(t)},type=numeric,method=classical[rk4],stepsize=
    0.1);
```
解を行列で取り出した例
```
> ff3:=dsolve({diff(x(t),t)=y(t),diff(y(t),t)=x(t)+y(t),x(0)=2,y
    (0)=1},{x(t),y(t)},type=numeric,output=array([0,.6,1.1,1.5,2.3,
    2.5,3.0]));
```
x(t) のデータを取り出します。
```
> ansmatrix:=ff3[2,1]:
> with(LinearAlgebra):
> #linalg[submatrix](ansmatrix,1..7,2..2);
> SubMatrix(Matrix(ansmatrix), 1..7, 2..2);
```
解を関数として取り出した例
```
> ff4:=dsolve({diff(x(t),t)=y(t),diff(y(t),t)=x(t)+y(t),x(0)=2,y
    (0)=1},{x(t),y(t)},type=numeric,output=listprocedure);
> fx:=subs(ff4,x(t)):fy:=subs(ff4,y(t)):
> fx(1);fy(1);
```
解をグラフにする方法

連立常微分方程式 $\frac{\mathrm{d}}{\mathrm{d}t}x(t)=y(t),\ \frac{\mathrm{d}}{\mathrm{d}t}y(t)=0.9\left(1-x(t)^2\right)y(t)-x(t)$ の解を様々な方法でグラフにします。
```
> restart;
```
plots **パッケージ** odeplot

dsolve と plots パッケージの odeplots を組み合わせて、２次元と３次元のグラフを作成します。
odeplot でグラフの範囲と描画点数を指定すると、それに合わせた計算刻みで計算を行い、グラフが表示されます。
```
> with(plots):
> p:=dsolve({diff(x(t),t)=y(t),diff(y(t),t)=0.9*(1-x(t)^2)*y(t)-x
    (t),x(0)=3,y(0)=2},{x(t),y(t)},type=numeric);
> odeplot(p,[t,x(t)],0..50,numpoints=150);
> odeplot(p,[t,x(t),y(t)],0..50,numpoints=150,color=black,axes=
    boxed);
```
DEtools **パッケージ** DEplot,DEplot3d

DEtools に直接、微分方程式を指定してグラフを作成する方法です。(アルゴリズムは Runge-KUtta 4次 を使用)
微分方程式と独立変数の範囲、初期値を指定した後、計算刻み(省略可能)やグラフ軸を指定します。
```
> with(DEtools):
> DEplot([diff(x(t),t)=y(t),diff(y(t),t)=0.9*(1-x(t)^2)*y(t)-x(t)
    ],[x(t),y(t)],t=0..50,[[x(0)=3,y(0)=2]],stepsize=0.1,scene=[t,x
    (t)]);
```
初期値を2種類指定した例
```
> DEplot([diff(x(t),t)=y(t),diff(y(t),t)=0.9*(1-x(t)^2)*y(t)-x(t)
    ],[x(t),y(t)],t=0..50,[[x(0)=3,y(0)=2],[x(0)=3.5,y(0)=2]],
    stepsize=0.1,scene=[t,x(t)],linecolor=[red,black]);
```
横軸を x(t) 、縦軸を y(t) に指定した例
(1次の連立微分方程式系でシステムが自励系の場合、位相平面には方向場も一緒に表示されます)
```
> DEplot([diff(x(t),t)=y(t),diff(y(t),t)=0.9*(1-x(t)^2)*y(t)-x(t)
    ],[x(t),y(t)],t=0..50,[[x(0)=3,y(0)=2]],stepsize=0.1,scene=[x
    (t),y(t)],linecolor=red,title="位相平面");
```
3次元プロットの例
```
> DEplot3d([diff(x(t),t)=y(t),diff(y(t),t)=0.9*(1-x(t)^2)*y(t)-x
    (t)],[x(t),y(t)],t=0..50,[[x(0)=3,y(0)=2]],stepsize=0.1,scene=
```

```
L   [t,x(t),y(t)],linecolor=black,title="位相平面");
```