

インタラクティブシステム論 数値計算ソフト SciLabに慣れる

梶本裕之
Twitter ID kajimoto
ハッシュタグ #ninshiki

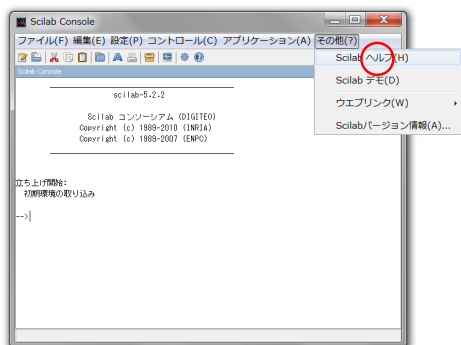
情報源



<http://www.scilab.org/>

web上:
 •SciLab入門(大野修一)
<http://www.eci.sys.hiroshima-u.ac.jp/scilab/introscilab/introscilab.html>
 •コマンド一覧
<http://www.eci.sys.hiroshima-u.ac.jp/scilab/man/ja/index-c.htm>
 •SciLabつかいませんか
<http://www-ec.denki.numazu-ct.ac.jp/scilab/index.html>

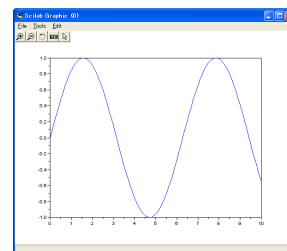
はじめの一步(1)立ち上げとヘルプの起動



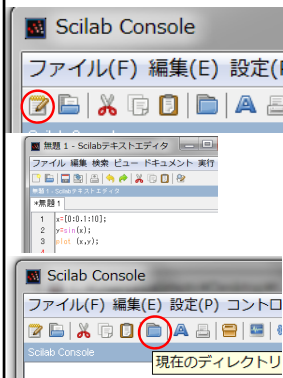
はじめの一步(2)プロットしてみる

コマンドプロンプトに一行ずつ打ち込む

```
x=[0:0.1:10];
y=sin(x);
plot(x,y);
```



はじめの一步(3)プログラムをファイルに保存



1. テキストエディタ起動
先ほどのスクリプトを入力
ファイル名を指定して保存
(ここではデスクトップに)
2. コンソールに戻る。
現在のフォルダを移動
現在のフォルダを確認
→`pwd`
ファイルがあることを確認
→`ls`
実行
→`exec('test.sce');`

プログラミングの基礎(1)

- 四則演算: `+`, `-`, `*`, `/`
- 1行に収まらない場合の表記 ...
(例) `11111+22222/33333-4444...
*55555`
- 計算結果の非表示: **行末に;** (;を付けなければ表示)
- プログラム中のコメント: `//`
- べき乗: `^`
(例) `x=3; y=2;z=x^y`
- 複素数: `%i`
- 共役複素数: `'`
(例)
`x=3+%i*2`
`y=x'`

例の実行結果を予想し、観察する

プログラミングの基礎(2)

•関係演算: `==(等)`, `!=(不等)`, `<`, `<=`, `>`, `>=(大小)`
 真の時%t, 偽の時%fの値をとる。
 (※ほとんどC言語と同じだが, 不等号は`!=`ではなく`~=`)
 (例)
`x=1; y=0;`
`x==y`
`x~=y`
`x>y`
 論理演算: `|`(論理和, or), `&`(論理積, and), `~`(否定)
 (例)
`x=1;y=1;z=2;`
`(x==y)&(y==z)`
`~(x==y)`
`(x>z)|(y==z)`

例の実行結果を予想し, 観察する

プログラミングの基礎(3) : 組み込み関数と定数

●組み込み関数: ヘルプの"Elementary Functions"を参照
 •三角関数: `sin`, `cos`, `tan`, `sind`, ..., `acos`, `asin`, `atan`, `acosd`, ... (dが付くと度)
 •平方根とべき乗: `sqrt`(平方根), `pow`(べき乗)
 •整数化: `round`(四捨五入), `floor`(切り下げ), `ceil`(切り上げ)
 •複素数関係: `real`(実部), `imag`(虚部), `angle`(位相角)
 •符号と絶対値: `abs`(絶対値), `sign`(符号)
 •商と余り: `mod`(商, 整数)
 •指数関数と対数関数: `exp`, `log`, `log10`, `log2`
 •その他の関数 `sinc`, `bessel`, などなどたくさん

●定数: %を付ける。
 •%i: 虚数単位
 •%pi: 円周率
 •%t: 真
 •%f: 偽

コンソールで `help sin` などとしてヘルプ画面を眺めること。

ベクトルと行列(1)

•行ベクトル
`x=[1,2,3,4]`
 •列ベクトル
`y=[1;2;3;4]` (;は改行を表す)
 •列ベクトルの別の表現
`y=[1,2,3,4]'` ('は行列の転置を表す. 行と列が交代する)

•等差数列からなる行ベクトル
`x=[1:10]`
`y=[1:10]'`
 •等差数列の差を指定出来る
`x=[1:0.2:4]`
`y=[0:2:10]'`

C言語などにおける配列はベクトル, 行列に統合

実行結果を観察する

ベクトルと行列(2)

•行列の指定
`A=[1,2,3`
`4,5,6]`

•零行列
`A=zeros(2,3)`
`B=zeros(1,3)`

•行列の指定の別の表現
`A=[1,2,3;4,5,6]` (;は改行)
 •成分がすべて1の行列
`A=ones(2,3)`

•ベクトルをくっつけて行列を作る
`x=[1,2,3]`
`A=[x;x]`
`B=[x;2*x;3*x]`
`C=[x,[4:6];6,5,4,4-x]`

•単位行列
`B=eye(3,3)`

実行結果を予想し, 観察する

ベクトルと行列(3)

•対角行列
`x=[1:3]`
`A=diag(x)`

•一様乱数行列(0から1の間)
`A=rand(2,3)`

•行列の加減算
`A=[1,2;3,4]`
`B=[5,6;7,8]`
`C=A+B`
`D=A+1` (行列の各成分に1を足す)

•行列の乗算
`A=[1,2;3,4]`
`B=[5,6;7,8]`
`C=A*B`
`D=A^2`

•ベクトル内積は行列乗算の一種
`x=[1,2]`
`y=[3,4]`
`z=x*y'`
`w=x'*y`
 (二つの結果が違うことに注意)

•行列の成分ごとの乗算
`C=A.*B`

•行列の転置
`D=A'`

実行結果を予想し, 観察する

ベクトルと行列(4)

•行列の割り算(逆行列を使う)
`A=[1,2;3,4]`
`x=[1,2]'`
`y=inv(A)*x` (`inv`は逆行列)

•正方行列のべき乗
`A=[1,2;3,4]`
`C=A^2`

•行列の成分ごとの割り算
`A=[1,2;3,4]`
`B=[3,4;1,2]`
`C=A./B`

実行結果を予想し, 観察する

ベクトルと行列(5)

•一般的な関数は成分ごとに効く

`x=[0:0.1:10]`

`y=sin(x)`

`z=abs(y)`

`plot(x,z)`

•ベクトルの長さ

`a=length(x)`

•ベクトルの平均, 合計

`a=mean(x)`

`a=sum(x)`

•行列のサイズ

`A=[1,2,3;4,5,6]`

`x=size(A)`

•行列の1成分を取り出す

`A=[1,2;3,4]`

`a=A(1,1)`

`b=A(1,2)`

•行列の成分を追加

`A(4,5)=1`

•行列の対角成分を取り出す

`A=[1,2;3,4]`

`c=diag(A)`

•行列の成分を取り出す

`A=[1,2,3;4,5,6]`

`a=A(1:2,1:2)`

`b=A(1,:) //:はすべてを示す`

`c=A(:,2)`

実行結果を予想し、観察する

行列の関数

•diag	行列式
•norm	ノルム
•orth	直交化
•rank	階数
•trace	トレース
•inv	逆行列
•pinv	一般化(疑似)逆行列
•spec	固有値と固有ベクトル
•expm	行列の指数関数
•logm	行列の対数関数
•sqrtm	行列の平方根
•lu	行列のLU分解
•qr	行列のQR分解

今はわからなくて構いません。他にもたくさん

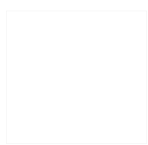
課題: 行列

次の行列の

(1) 逆行列を求め

(2) 逆行列と元の行列をかけると単位行列になることを確認

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$$



2次元グラフ(1)

●plot(x,y)

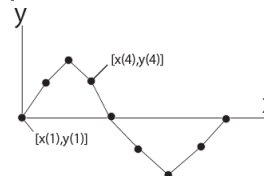
x軸にベクトルx,y軸にベクトルyを対応させてプロットする。

`x=[0:0.1:10];`

`y=sin(x);`

`plot(x,y);`

$[x(1), y(1)], [x(2), y(2)], \dots, [x(N), y(N)]$ をつないでグラフにする。



2次元グラフ(2): 書式

- plot(x1,y1,x2,y2,...) 複数のグラフを書く
- xtitle('title', 'xlabel', 'ylabel') タイトルとラベルを付ける
- legend('leg1', 'leg2',...) 凡例を付ける
- xgrid グリッド線を入れる
- grid on グラフ中にグリッドを入れる

`x=[0:0.1:10];`

`y=sin(x);`

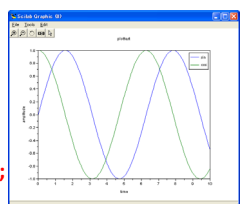
`z=cos(x);`

`plot(x,y,x,z);`

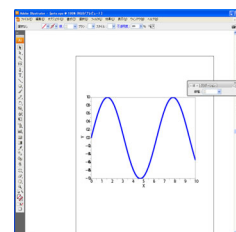
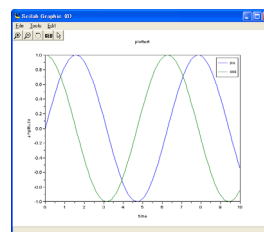
`legend('sin','cos');`

`xtitle('plottest','time','amplitude');`

`xgrid();`



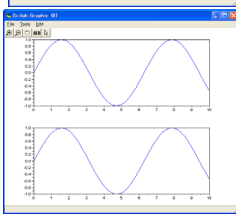
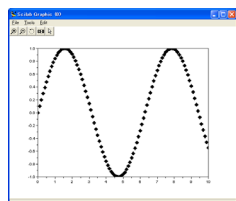
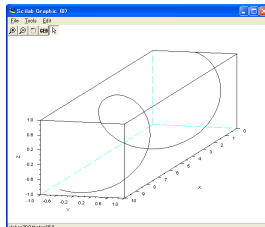
グラフの保存



- 「ファイル」⇒「エクスポート」でファイルに保存可能。
- png, jpeg等, 用途に応じて。
- epsであればIllustrator等で修正可能

グラフ:その他いろいろ

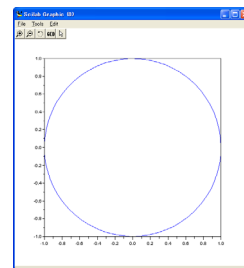
- 線の太さ
- 線の種類
- 複数のグラフの描画
- 3次元グラフ
(必要に応じてヘルプ参照)



課題:円を描く

- plot関数を使って円を描いてみる

```
rad = [0:0.1:2*pi];
x=cos(rad);
y=
plot(x,y);
```



レポート課題(1):リサージュ図形



「リサージュ図形」について調べ、
plot関数で描いてみよう

制御文(1)

```
*for文 (for...end)
x=0;
for i=1:10 //ベクトルと同じ表記法. i=1:0.1:10だと?
    x=x+1; //C言語のような+=は無い
end
x //表示

*while文 (while...end)
i=10; x=0;
while i>0
    x = x+i;
    i = i-1;
end
x
```

実行結果を予想し、観察する

制御文(2)

```
*if文 (if...elseif...else...end)
x=[0:0.1:10]; //ベクトルを定義
y=[]; //空のベクトルを定義
for i=1:length(x) //これの意味は?
    y(i)=sin(x(i));
    if y(i)<0
        y(i)=0;
    elseif(y(i)<0.5)
        y(i)=0.5;
    else
        y(i)=1.0;
    end
end
plot(x,y) //表示
```

実行結果を予想し、観察する

※C言語のif文では“else if”だったが、Scilabでは“elseif”

制御文(3)

```
select文 (select...case...case...else...end)
for i=-3:3
    select sign(i)
        case 1
            printf('%d is positive¥n',i);
        case -1
            printf('%d is negative¥n',i);
        else
            printf('%d is zero¥n',i);
    end
end
```

実行結果を予想し、観察する

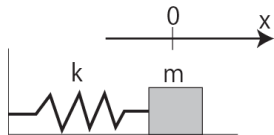
制御文: 授業中課題

ばねの挙動をシミュレートしたい。
四角の中はどうか? 考え, 実行せよ。

```

m=1.0; //重さ
k=1.0; //ばね定数
x=1.0; //初期位置
v=0; //初期速度
dt=0.1; //時間刻み
record=[]; //記録用
for time= 0:dt:10 //時刻
    F=-k*x; //ばねによって生じる力
    a=F/m; //生じる加速度
    v= v+dt*a; //速度
    x= x+dt*v; //位置
    record = [record,x]; //記録(*テクニック:ベクトルが伸びていく)
end
plot([0:dt:10],record);

```



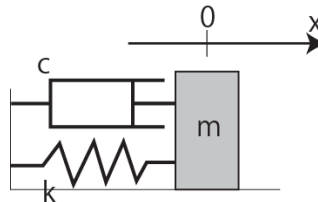
レポート課題(2)

速度に比例したブレーキ(ダンパ, 粘性)が加わったとき
の様子をシミュレートせよ。

(ヒント: 力の部分が変わる)

バネマス系: $F=ma=-kx$

バネマスダンパ系: $F=ma=-kx-cv$



関数(1)

- 関数はメインプログラムとは別のファイル。
- メインプログラムではexec('')によって関数ファイルを読み込む

✓関数ファイル myfunc.sceの中身

```

function out = myaverage(x) //out: 返り値. x:引数
    y = sum(x); //sum:ベクトル, 行列の合計
    y = y / length(x); //length:ベクトルの長さ
    out = y; //返り値に代入
endfunction //最後はendfunction

```

✓メインプログラム test.sceの中身

```

exec('myfunc.sce'); //関数ファイルの読み込み
x=[1:10];
y=myaverage(x)

```

実行結果を予想し, 観察する

関数(2)

- 関数の引数, 返り値はベクトル, 行列でもよい。
- 返り値は複数でもよい

✓関数ファイル myfunc.sceの中身

```

function [out1,out2] = myaverage(x,y)
    out1 = x+y;
    out2 = x-y;
endfunction

```

✓メインプログラム test.sceの中身

```

exec('myfunc.sce');
time=[0:0.1:10];
x=sin(time);
y=cos(time);
[a,b]=myaverage(x,y);
plot(time,a,time,b);

```

実行結果を予想し, 観察する

音の扱い(1)

●playsnd(y) 系列yを鳴らす. yは横ベクトル

```

x=[0:0.1:1000];
y=sin(x); //1 × 10000の正弦波
playsnd(y);

y=sin(2*x); //高い正弦波
playsnd(y);

y=[sin(x); sin(4*x)]; //2 × 10000の正弦波
playsnd(y); //右耳と左耳で違う音が聞こえる

y=rand(1,10000); //ランダム系列
playsnd(y); //ホワイトノイズが聞こえる

```

実行結果を予想し, 観察する

音の扱い(2)

- loadwave('filename') waveファイルを読み込む
- savewave('filename',y) 系列yをwaveファイルにして書き出す

```

y=[sin(x); sin(4*x)]; //2 × 10000の正弦波
playsnd(y); //右耳と左耳で違う音が聞こえる
savewave('sample.wav',y);

```

※ファイルの保存場所はpwdコマンドによって確認する
※保存する場所を変えなければcdコマンドを用いる。

出来たwaveファイルを再生してみる

レポート課題(3): 余裕のある場合のみ

簡単な音楽を作成せよ。提出はwaveファイルではなくScilabのプログラムファイルが良い。

(ヒント)ドレミファソラシドの各周波数を調べる。

レポート課題

(1)(2): 必須、(3): 時間あれば
レポートは授業で指定した場所に提出。
Scilab(またはPython)のプログラムを添付