c1/07-08.tex, 2018.05.10

1 一変数代数方程式の求解のコマンド

```
gp (unix shell より pari/gp を起動)
polroots(x^3-x-1)
quit  (unix shell へ戻る)
```

```
openxm fep asir (unix shell より asir を起動)
pari(roots,x^3-x-1);
quit(); (unix shell へ戻る)
```

2 asir での Newton 法のプログラム sqrt2.rr

MathLibre または orange2b で,

```
openxm use-asir-mode.sh --local-yes
```
(.emacs の asir 用設定, 一度だけ. MathLibre ではすでに誰かがやってあれば不要)

```
openxm emacs sqrt2.rr &
```

以下を入力すると asir メニューから実行できる.

```
import("names.rr")$
def mysqrt(A) {
  A = eval(A*exp(0));
  X = A;
  for (I=0; I<5; I++) {
    Y = (X+A/X)/2;
/*    print(Y); */
    X = Y;
  }
  return(Y);
}
mysqrt(2);
end$
```

3 二変数 Newton 法の asir でのプログラム p-0511.rr

```
import("names.rr")$
def jac(X,Y) {
  return newmat(2,2,[[2*X,2*Y],[Y,X]]);
}
def ff(X,Y) {
  return newvect(2,[X^2+Y^2-4,X*Y-1]);
}
def newton2(X0,Y0) {
  P=newvect(2,[X0,Y0]);
  for (I=0; I<5; I++) {
   P = P - matrix_inverse(jac(P[0],P[1]))
      *ff(P[0],P[1]);
   printf("P=%a,ff(P)=%a\n",P,ff(P[0],P[1]));
  }
  return P;
}
newton2(2.0,0.1);
//初期値を [2.0, 0.1] として Newton method
end$
```

4 Mathematica

4.1 一変数代数方程式の求解のコマンド

```
math (unix shell より Mathematica を起動. orange3m, 2 名のみ)
N[Solve[x^3-x-1==0,{x}]]
Quit   (unix shell へ戻る)
```

参考: <<JavaGraphics‘ (orange3m の math で Graphic の表示をしたい場合)

4.2 Mathematica での Newton 法のプログラム sqrt2.m

math で起動[1]. emacs でプログラムを書いて <<sqrt2.m で実行.

```
mysqrt[aa_]:=Block[{x,i,y,a},
  a = N[aa];
  x = a;
  For[i=0, i<5, i++,
    y = (x+a/x)/2;
    x = y;
  ];
  Return[y];
]
```

[1] orange3m のみ. 2 名まで