```
> restart;
```

Procedury a funkce

```
> F:=(x)->sin(x)*exp(-x);
```

$$F := x \to \sin(x)\, \mathbf{e}^{(-x)}$$

Co když chceme udělat funkci z něčeho, co jsme již pracně spočetli?

```
> P4:=normal(eval(subs(n=4,1/2^n/n!*diff((x^2-1)^n,x$n))));
```

$$P4 := \frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}$$

```
> P4_spatne:=(x)->P4;
```

$$P4\_spatne := x \to P4$$

```
> P4_spatne(z);
```

$$\frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}$$

Proto tu je **unapply**

```
> P4_lepsi:=unapply(P4,x);
```

$$P4\_lepsi := x \to \frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}$$

Podobně

```
> normal(eval(1/2^n/n!*diff((x^2-1)^n,x$n)));
```

$$\frac{\mathrm{diff}((x^2-1)^n, x\,\$\,n)}{2^n\, n!}$$

```
> PP:=unapply(%,n,x);
```

$$PP := (n,x) \to \frac{\mathrm{diff}((x^2-1)^n, x\,\$\,n)}{2^n\, n!}$$

```
> PP(3,x);
```

$$x^3 + \frac{3}{2}(x^2-1)\,x$$

```
> PP(3,cos(theta));
Error, (in PP) wrong number (or type) of parameters in function diff
>
>
```

Nakonec se tedy budeme mustet snžit k "programovn"

```
>
> P:=proc(n,x)
    local y;

    if n=0 then
            1
    else
            normal(subs(y=x,eval(1/2^n/n!*diff((y^2-1)^n,y$n))))
    fi;
  end;
```

```
P := proc(n, x)
local y;
    if n = 0 then 1 else normal( subs( y = x, eval( diff( (y^2 - 1)^n, y $ n ) / (2^n*n! ) ))) fi
end
```

I zde jsou speciality, jako třeba implictn nvratov hodnota dan poslednm provedenm vrazem před koncem běhu procedury

```
> P(2,cos(theta));
```

$$\frac{3}{2}\cos(\theta)^2 - \frac{1}{2}$$

Jenže

```
> P(n,cos(theta));
  simplify(subs(n=3,%));
```

$$\frac{\mathrm{diff}((\cos(\theta)^2 - 1)^n, \cos(\theta)\ \$\ n)}{2^n\, n!}$$

```
Error, (in simplify) wrong number (or type) of parameters in function indets
> 
```

Musme se tedy při provděn procedury ujistit, že parametry jsou již dostatečně znmy.
A v přpadě, že ne, MUSÍME vrtit nevyhodnocen vraz rovn pvodnmu voln fukce.
Tak zařdme, aby až bude znm přslučn parametr zavolal explicitn či implictn eval funkci znovu.

```
> P:=proc(n,x)
    local y;

    if not type(n,integer) then RETURN('P(n,x)'); fi;
    if n=0 then
            RETURN(1)
    else

  RETURN(normal(subs(y=x,eval(1/2^n/n!*diff((y^2-1)^n,y$n)))))
    fi;
  end;
P := proc(n, x)
local y;
    if not type( n, integer ) then RETURN( 'P( n, x )' ) fi;
    if n = 0 then RETURN( 1 )
    else RETURN( normal( subs( y = x, eval( diff( (y^2 - 1)^n, y $ n ) / (2^n*n! ) ))))
    fi
end
> P(n,cos(theta));
  simplify(subs(n=3,%));
```

$$\mathrm{P}(n, \cos(\theta))$$

$$\frac{5}{2}\cos(\theta)^3 - \frac{3}{2}\cos(\theta)$$

```
> 
```

Mžeme tak deklarovat použit globlnch proměnnch;
Rozsah platnosti je stejn jako v běžnch jazycch

```
>
> Q:=proc(a,b,c)
   local l,m,n;
   global g,h,i;

   #body

  end;
```

$$Q := \textbf{proc}(a, b, c)\ \textbf{local}\ l, m, n;\ \textbf{global}\ g, h, i;\ \ \textbf{end}$$

```
>
```

Je tu ale jeden detail, aby se nepltvalo časem, vyhodnocuj se lokln proměnn jen do hloubky 1

```
> Q:=proc()
     local A,x; global n;

     A:=diff((x^n),x);
     n:=3;
     RETURN(A);
  end;
```

$$Q := \textbf{proc}()\ \textbf{local}\ A, x;\ \textbf{global}\ n;\ n := 'n';\ A := \mathrm{diff}(x\hat{}n, x);\ n := 3;\ \mathrm{RETURN}(A)\ \textbf{end}$$

```
> Q();
  eval(%);
  n:='n':   #    klid
```

$$\frac{x^n\, n}{x}$$

$$3\, x^2$$

Dovnitř již deklarovan procedury mžeme nahldnout

```
> print(LegendreP);
```

$$\textbf{proc}(a1{::}algebraic, a2{::}algebraic, a3{::}algebraic)\ \ldots\ \textbf{end}$$

```
> interface( verboseproc = 3 );#  Tohle se nevrt do pvodnho stavu pomoc restart !
> print(LegendreP);
```

$$\textbf{proc}(a1{::}algebraic, a2{::}algebraic, a3{::}algebraic)\ \ldots\ \textbf{end}$$

```
>
>
```

Podobně jako v archaickch jazycch si mžeme nechat vypsat kroky, kter procedura provd

```
> restart;
  printlevel:=10;
  int(x^2,x);
  printlevel:=1;
```

$$printlevel := 10$$

```
{--> enter int, args = x^2, x
{--> enter int/int, args = [x^2, x], 10, _EnvCauchyPrincipalValue
```

$$answer := [\tfrac{1}{3} x^2,\ x,\ [formula]]$$

$$\frac{1}{3}x^3$$

```
<-- exit int/int (now in int) = 1/3*x^3}
```

$$answer := \frac{1}{3}x^3$$

$$\frac{1}{3}x^3$$

```
<-- exit int (now at top level) = 1/3*x^3}
```

$$\frac{1}{3}x^3$$

$$printlevel := 1$$

```
>
```

Ještě je tu dalš specialita:

```
> fib:=proc(n)
   if   n=0 then 0
   elif n=1 then 1
   else fib(n-1)+fib(n-2)
   fi;
  end;
```

$$fib := \mathbf{proc}(n)\ \mathbf{if}\ n = 0\ \mathbf{then}\ 0\ \mathbf{elif}\ n = 1\ \mathbf{then}\ 1\ \mathbf{else}\ \mathrm{fib}(n-1) + \mathrm{fib}(n-2)\ \mathbf{fi}\ \mathbf{end}$$

```
> fib(27);
```

$$196418$$

Vpočet trv tak dlouho, protože 2^26 je 67 milion

```
>
> fib:=proc(n)
   if   n=0 then 0
   elif n=1 then 1
   else fib(n):=fib(n-1)+fib(n-2)
   fi;
  end;
```

$$fib := \mathbf{proc}(n)\ \mathbf{if}\ n = 0\ \mathbf{then}\ 0\ \mathbf{elif}\ n = 1\ \mathbf{then}\ 1\ \mathbf{else}\ \mathrm{fib}(n) := \mathrm{fib}(n-1) + \mathrm{fib}(n-2)\ \mathbf{fi}\ \mathbf{end}$$

```
>
> fib(27);
```

$$196418$$

Nyn jsme využili vnitřn tabulku funkčnch hodnot, kterou si Maple vede pro zapamatovn již spočtench hodnot

```
> op(fib);
```

$$\mathbf{proc}(n)\ \mathbf{if}\ n = 0\ \mathbf{then}\ 0\ \mathbf{elif}\ n = 1\ \mathbf{then}\ 1\ \mathbf{else}\ \mathrm{fib}(n) := \mathrm{fib}(n-1) + \mathrm{fib}(n-2)\ \mathbf{fi}\ \mathbf{end}$$

```
>
> restrat;
  fib:=proc(n)
   option remember;
   if   n=0 then 0
   elif n=1 then 1
   else fib(n-1)+fib(n-2)
```

```
    fi;
  end;
```

restrat

fib :=

proc(*n*) **option** *remember*; **if** *n* = 0 **then** 0 **elif** *n* = 1 **then** 1 **else** fib(*n* − 1) + fib(*n* − 2) **fi end**

```
> fib(27);
```

196418

```
>
> P:=proc(n)
    local u,x;
    option remember;
    if (n=1) then (x)->x
    elif n=0 then (x)->1
    else
     u := simplify((2*n-1)/n*P(n-1)(x)*x-(n-1)/n*P(n-1)(x));
     unapply( u, x);
    fi;
  end;
```

P := **proc**(*n*)
local *u*, *x*;
option *remember*;
  if *n* = 1 **then** *x* → *x*
  elif *n* = 0 **then** 1
  else
    u := simplify(((2∗*n* − 1)∗P(*n* − 1)(*x*)∗*x*) / *n* − ((*n* − 1)∗P(*n* − 1)(*x*)) / *n*);
    unapply(*u*, *x*)
  fi
end

```
> P(24)(cos(theta)):
>
```

Knihovny

když umme pst funkce, mohli bychom začt pst moduly. Kdyby to snad někdy někdo potřeboval ať si přečte npovědu k **realib,writelib.**

```
>
```

Z historickch dvod se kromě **library** použv mocnější **package**. Plat, že větš, universlnější věci se bal do balk, menš speciln do knihoven.

```
>
> with(plots);
```

[*animate*, *animate3d*, *animatecurve*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *cylinderplot*, *densityplot*, *display*, *display3d*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *odeplot*, *pareto*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*,

polyhedra_supported, *polyhedraplot*, *replot*, *rootlocus*, *semilogplot*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *sphereplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*]

```
> readlib(mtaylor);
```

proc() ... **end**

Nejdřve ten **mtaylor**, knihovna pro prci s rozvoji funkc vce proměnnch

```
>
> mtaylor(sin(x-y),{x,y,z});
```

$$x - y - \frac{1}{6}x^3 + \frac{1}{2}y\,x^2 - \frac{1}{2}y^2\,x + \frac{1}{6}y^3 + \frac{1}{120}x^5 - \frac{1}{24}y\,x^4 + \frac{1}{12}y^2\,x^3 - \frac{1}{12}y^3\,x^2 + \frac{1}{24}y^4\,x - \frac{1}{120}y^5$$

```
> whattype(%);
```

+

```
>
```

Nyn vybran funkce z balku **plots**

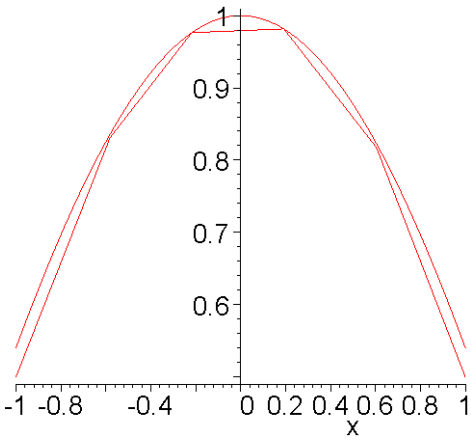
```
>
```

předevšm **display** zobraz uschovan vsledek malovn

```
>
> A:=plot(1-x^2/2,x=-1..1,numpoints=6,adaptive=false);
  B:=plot(cos(x),x=-1..1):
```

A := PLOT(CURVES([[-1., .5000000000000000],
[-.5814946400000001, .8309319918256352], [-.2173553639999999, .9763783228702138],
[.1921573360000000, .9815377791106915], [.6043845520000002, .8173596566518795],
[1., .5000000000000000]], COLOUR(*RGB*, 1.0, 0, 0)), AXESLABELS("x",),
VIEW(-1. .. 1., *DEFAULT*))

```
> display(A,B);
```



Jak je vidět, použv Maple k uschovn obrzk repreznatci zznamu pomoc symbolickch funkc, jejich seznam zjistme:

```
> ?plot/structure
```

a ještě se jim budeme věnovat

```
>
```

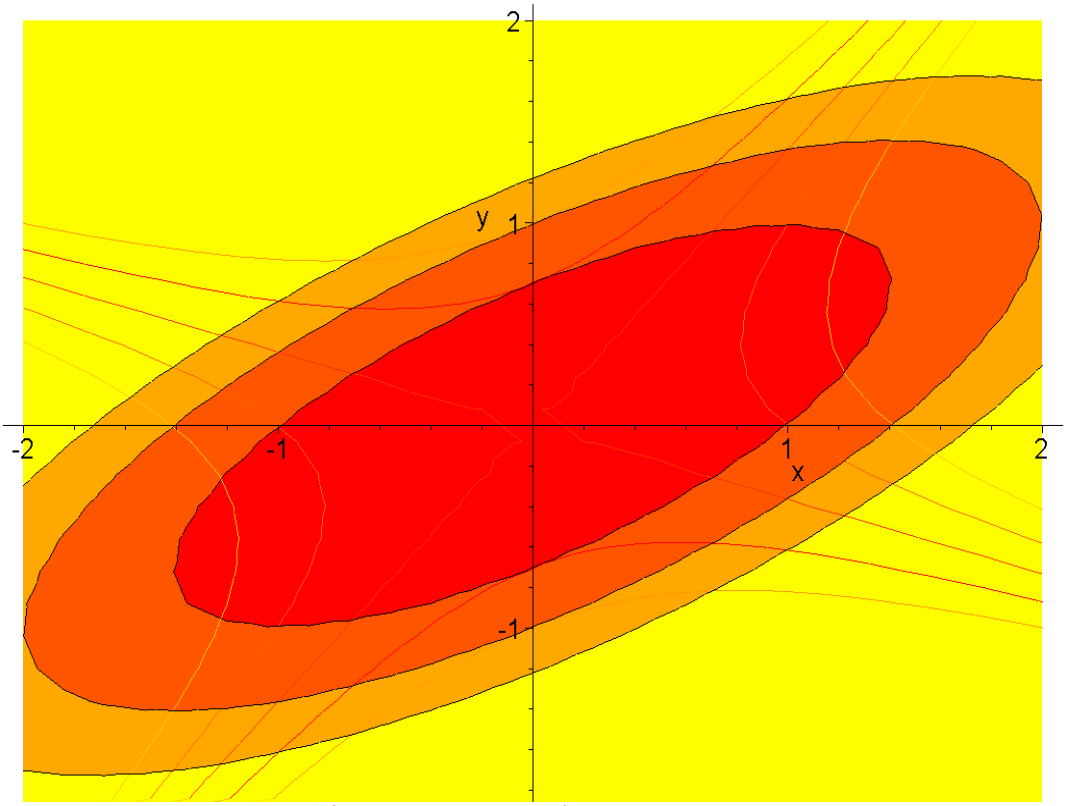
Mnoho funkci z balku **plots** jsou jen zkratky za něco, co bychom asi zvldli sami

```
> complexplot(arctan(x-I/2),x=-1..1);
  # je zkratka za
```

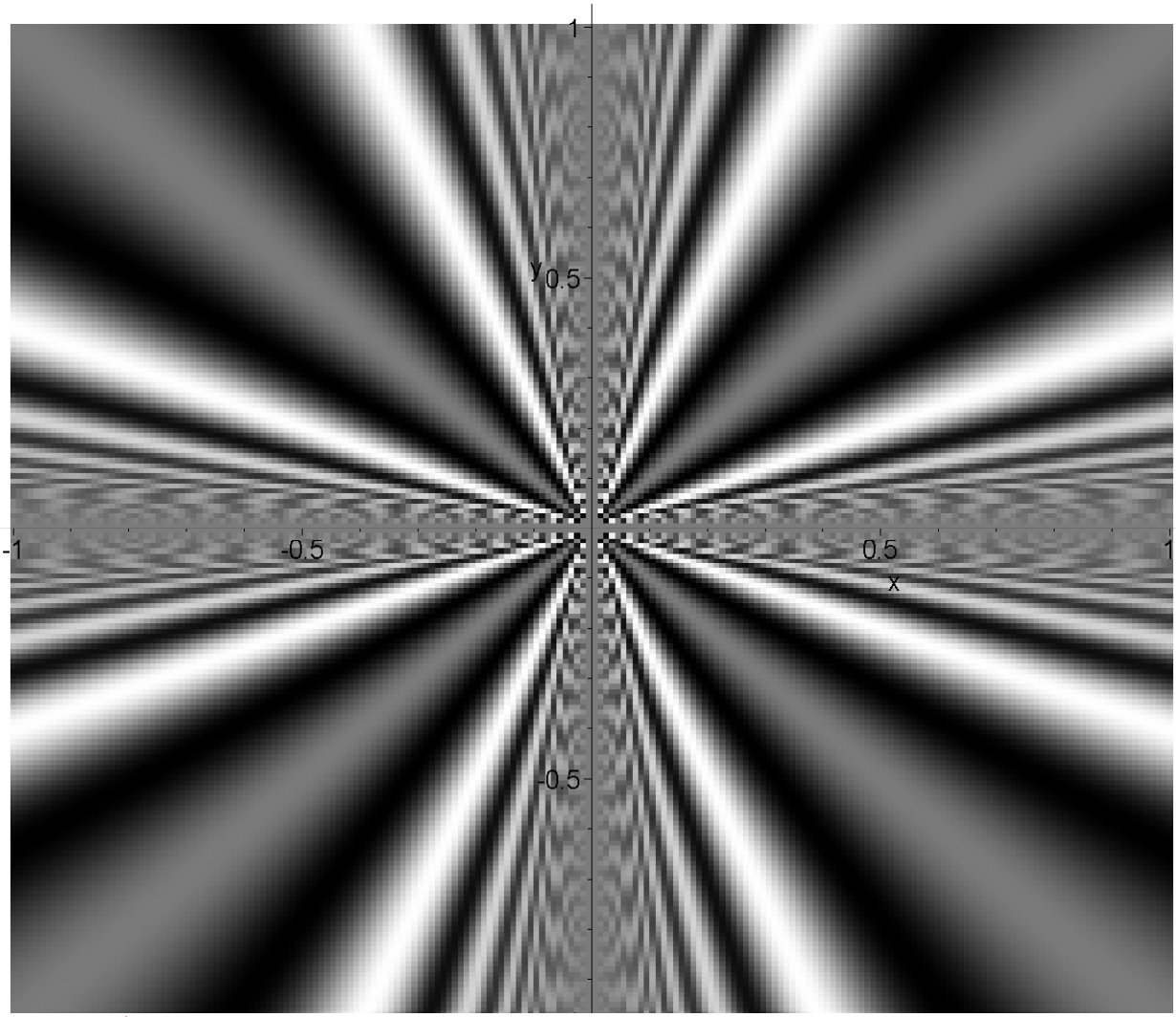
```
> plot([Re(arctan(x-I/2)),Im(arctan(x-I/2)),x=-1..1]);
```

Nikoli ale **contourplot, densityplot, ...**

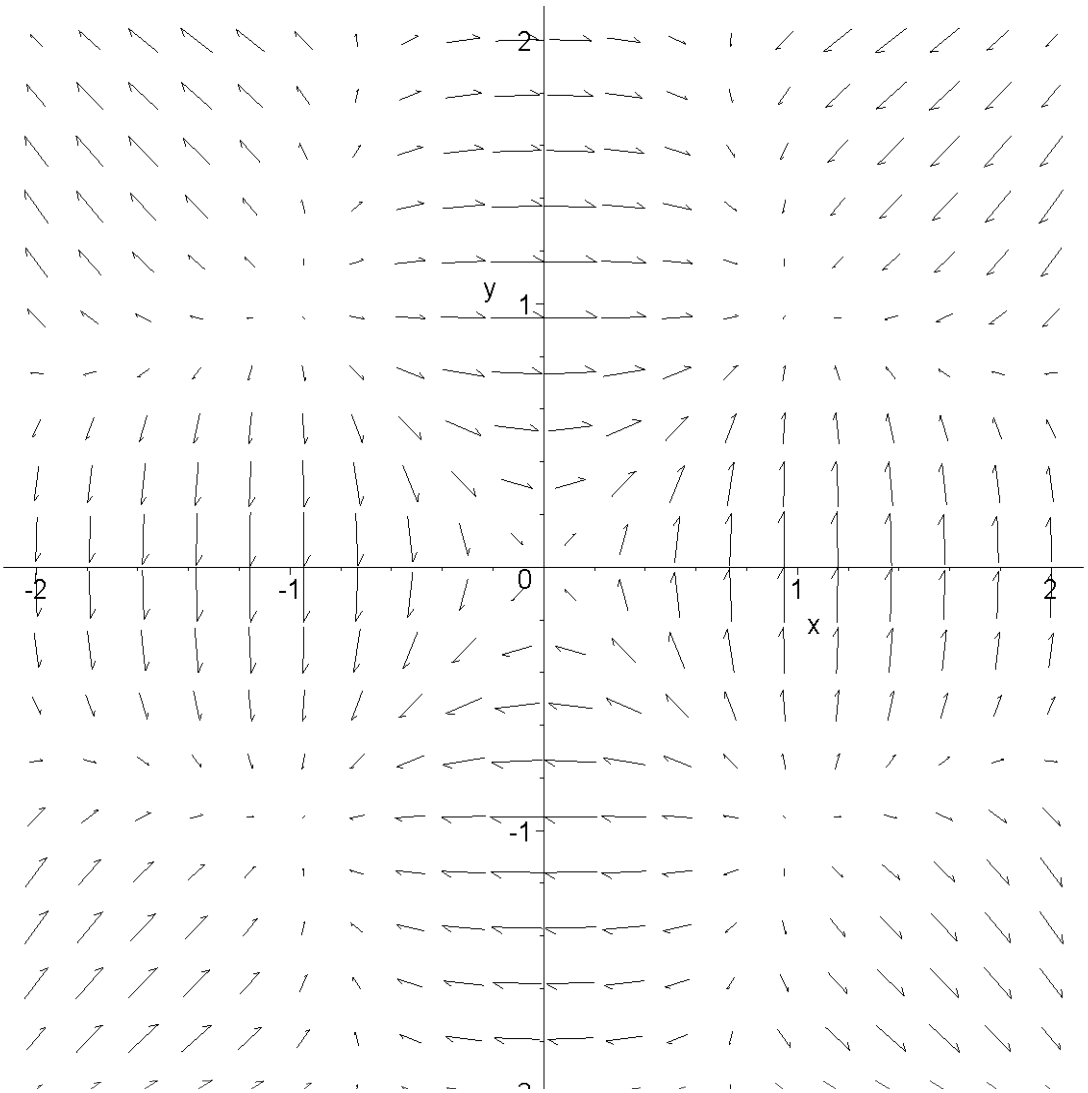
```
>
> A:=contourplot(
  x^2-2*y^2+2*x*y,x=-2..2,y=-2..2,levels=[-1,0,1,-2,2]):
  B:=contourplot(
  x^2+2*y^2-2*x*y,x=-2..2,y=-2..2,levels=[1,2,3],filled=true,color
  ing=[red,yellow]):
  display(B,A);
```



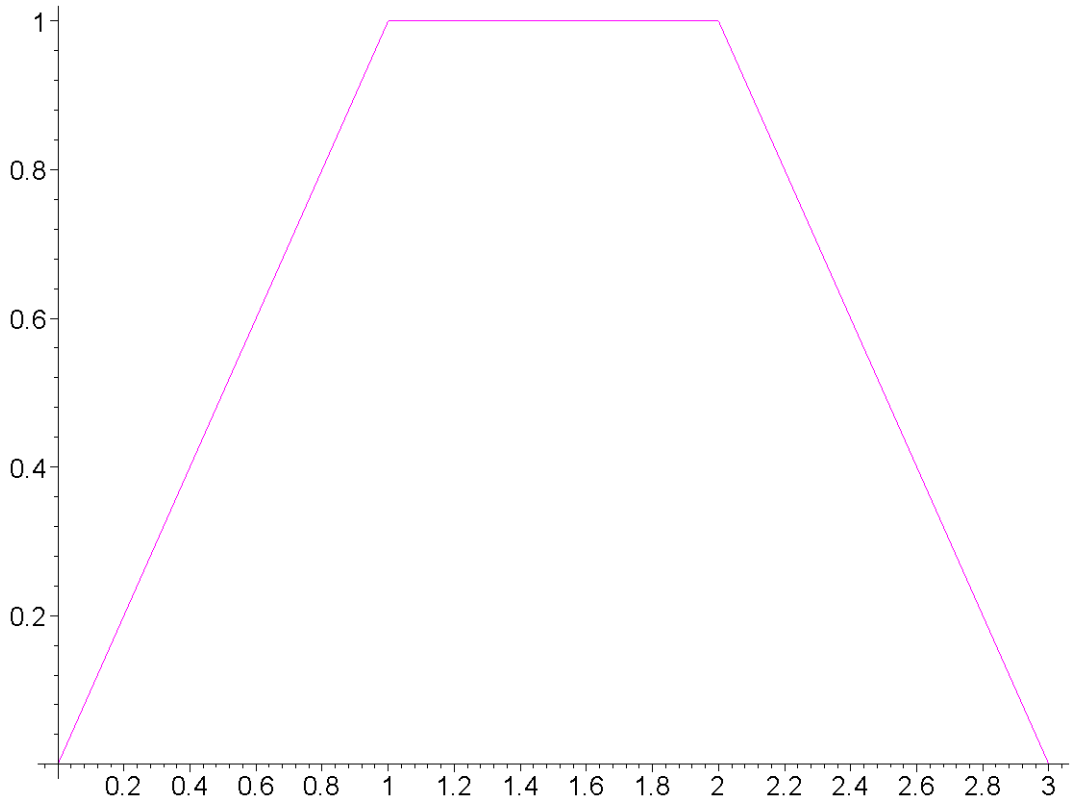

```
> densityplot(sin(Pi*x/y)*sin(Pi*y/x),x=-1..1,y=-1..1,grid=[200,20
  0],style=PATCHNOGRID);
```




```
> ?plot/options
>
>
>
> gradplot( sin(x*y)/(x^2+y^2+1),x=-2..2,y=-2..2);
```




```
> #fieldplot( [y,-x] ,x=-2..2,y=-2..2);
> #fieldplot(
  map(t->diff('sin(x*y)/(x^2+y^2+1)',t),[x,y]),x=-2..2,y=-2..2);
>
> display(
  PLOT(CURVES([[0,0],[1,1],[2,1],[3,0]]),COLOR(RGB,1,0,1)) );
```




```
>
```

Domc loha: modifikujte nsledujc řdky tak aby to kreslilo něco hezčho

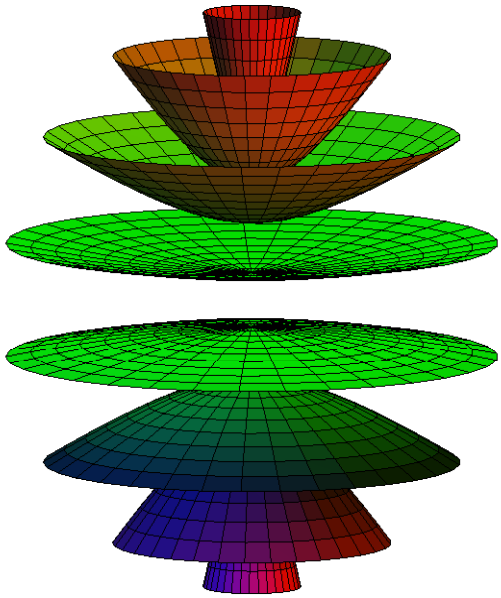
```
>
> Nf:=33: Nt:=8: Nr:=10:
  q:=1.8:
  sfs:=(u,theta,phi)->([u*sin(theta)*cos(phi),
                        u*sin(theta)*sin(phi),
                        sqrt(u^2+1)*cos(theta)]);
  K1:=(r,t,f)->evalf(sfs((q*r/Nr),t/Nt*Pi,f/Nf*2*Pi)):
  K:=(r,t,f)->[K1(r-1,t,f-1),K1(r,t,f-1),K1(r,t,f,0),K1(r-1,t,f)]:


  S:=[ seq(seq(seq(K(u,t+1/2,f),f=1..Nf),t=0..Nt-1),u=1..Nr) ]:
```

$$sfs := (u, \theta, \phi) \rightarrow [u \sin(\theta) \cos(\phi), u \sin(\theta) \sin(\phi), \sqrt{u^2 + 1} \cos(\theta)]$$

```
>
> P:=PLOT3D(POLYGONS(op(S)),
        LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),
        LIGHT(100,-45,0.0,0.0,0.7), AMBIENTLIGHT(0.2,0.2,0.0),
        STYLE(PATCH),COLOR(ZHUE),SCALING(CONSTRAINED)):

> display(P);
```



```
> 
> 
> 
> 
> 
> 
> 
> 
> 
```