Analysis 3D Rotations-Paraboloid- der Hut

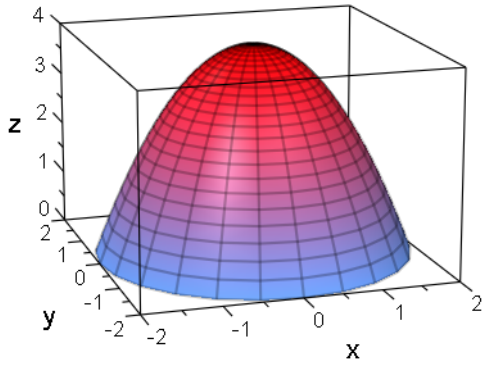
Mathematik mit MuPAD 2.5, Prof. Dr. Dörte Haftendorn 12.11.03 Version vom 12.11.03
MuPAD 4, Verbessert 2012, inzwischen sind die Grafik-Dateien auch nicht mehr groß.

LLLLLLLLL-------Level 1-------LLLLLLLLL

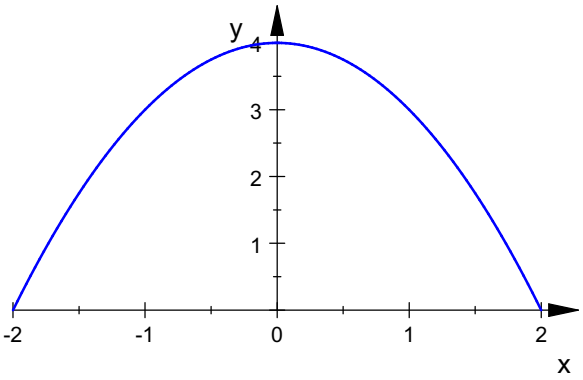
Veraltet: Ein Notebook mit 3D-Graphen ist riesig, 5 MB, bzw 16 MB!!!,wenn die Graphen verfeinert sind.
Da das ganz unakzeptabel ist, sind im Notebook die Ausgaben gelöscht (mit Notebook/Ausgaben löschen).
Nun muss man mit Notebook/Evaluiere/alle Eingaben alles neu auswerten.

War nun kommt ist zeitlos lohnend

Eine nach unten geöffnete Parabel soll sich drehen




```
plotfunc2d(4-x^2,x=-2..2)
```

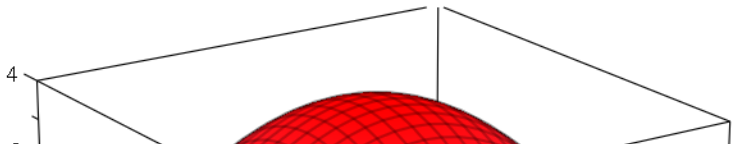


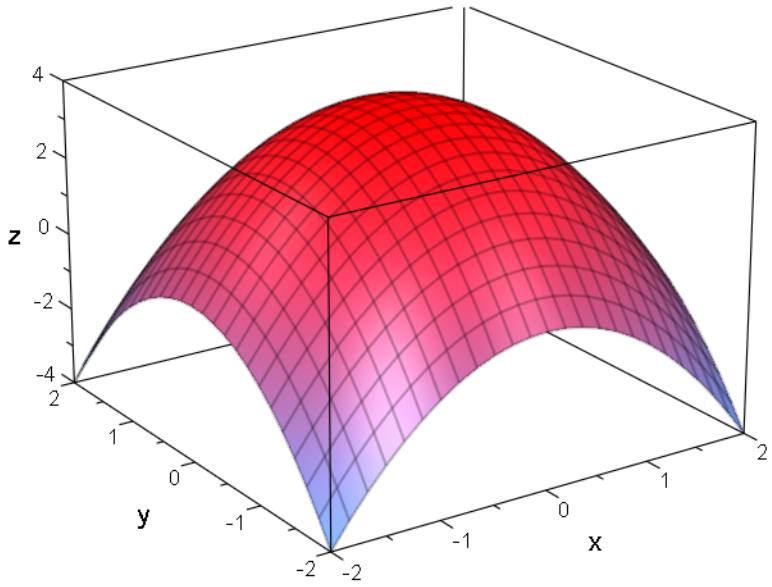
Die Hochachse soll die z-Achse sein. Dann sehen wir hier die x-z-Ebene.
In der y-z-Ebene muss es genauso aussehen. Also:

```
f:=(x,y)->(4-x^2-y^2);
```

$$(x,\ y) \to 4 - x^2 - y^2$$

```
plotfunc3d(f(x,y),x=-2..2,y=-2..2);
```



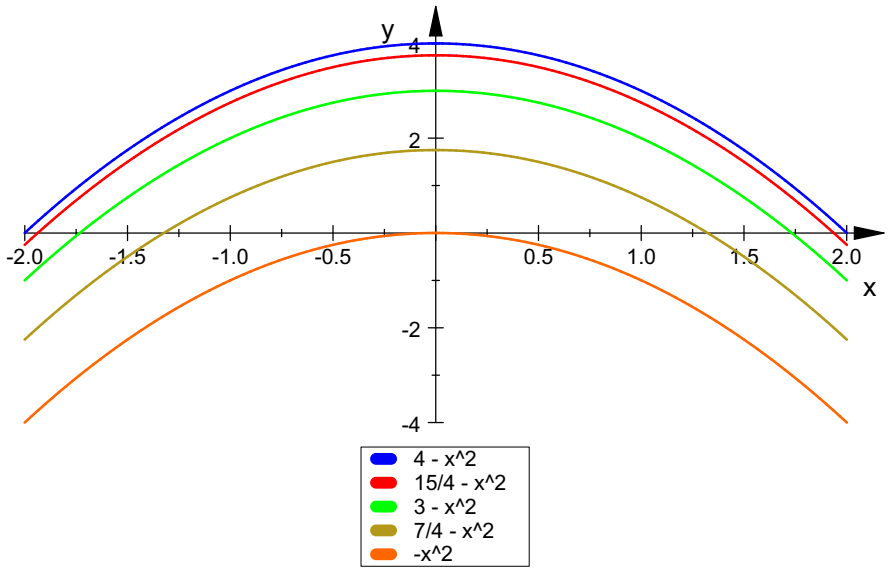
Reinklicken, frei drehen, im Objektbrowser rechts ggf. Objekteigenschaften ändern.

Schnitte senkrecht zur y-Achse, also y=konstant

```
schnitte_y_c:=(f(x,k/2) $ k=0..4)
```

$$4 - x^2,\ \frac{15}{4} - x^2,\ 3 - x^2,\ \frac{7}{4} - x^2,\ -x^2$$

```
plotfunc2d(schnitte_y_c,x=-2..2)
```

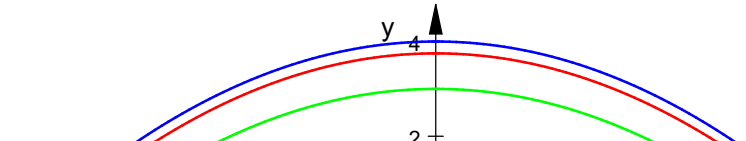


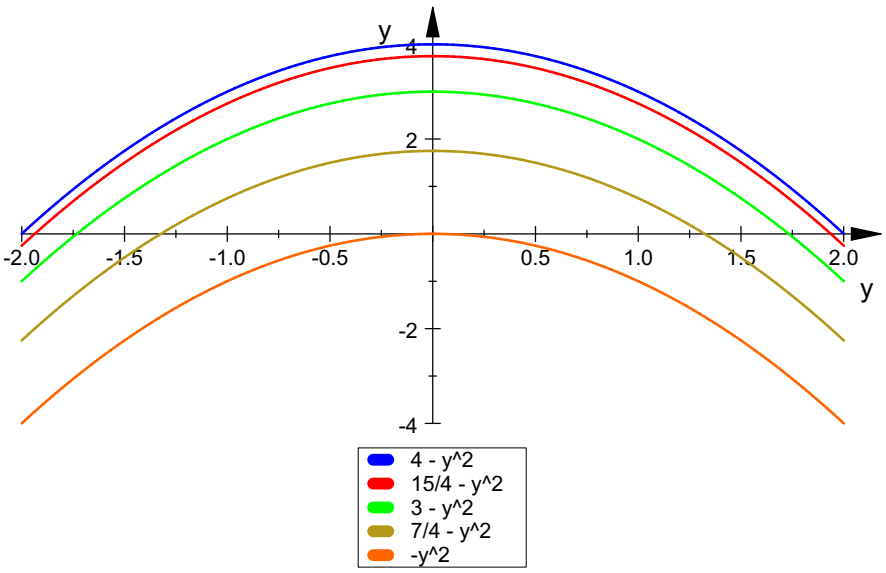
Schnitte senkrecht zur x-Achse, also x=konstant

```
schnitte_x_c:=(f(k/2,y) $ k=0..4)
```

$$4 - y^2,\ \frac{15}{4} - y^2,\ 3 - y^2,\ \frac{7}{4} - y^2,\ -y^2$$

```
plotfunc2d(schnitte_x_c,y=-2..2)
```



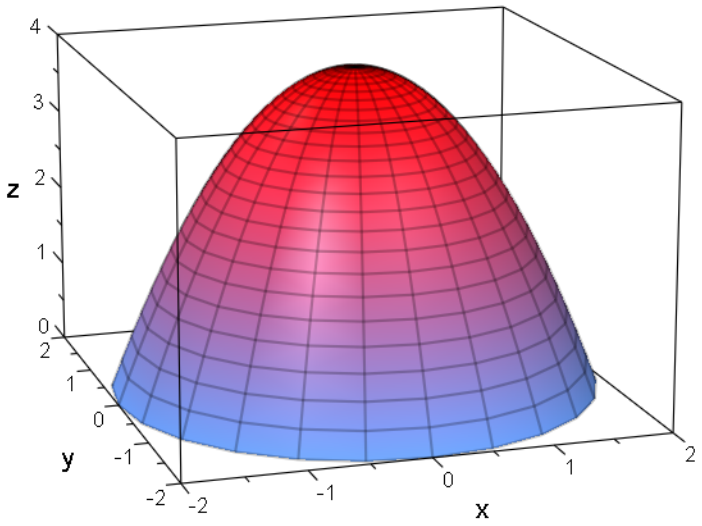
Wie erwartet sind alles verschobene Parabeln.
Es sind die Gitterlinien der 3D-Graphik.

LLLLLLLLL-------Level 2-------LLLLLLLLL

Eigentlich möchte man aber auch die Kreise sehen, die durch die Rotation entstehen.

Dazu muss man für x und y Polarkoordinaten nehmen: **x=r cos (phi) und y=r sin (phi) .**
Für z nimmt man den ursprünglichen Funktionsterm, geschrieben mit r, also **z = f (r).**
Nun braucht man das Werkzeug für Raumflächen. Zuerst wird ein **Graphik-Objekt** mit dem Namen "rund" erzeugt, das dann von der universellen Funktion **plot** dargestellt wird.

```
rund := plot::Surface(
  [r*cos(phi), r*sin(phi),4-r^2], phi = 0..8*PI/4, r =0..2):
plot(rund);
```



Nun sind die Kreise als die Höhenlinien zu sehen.

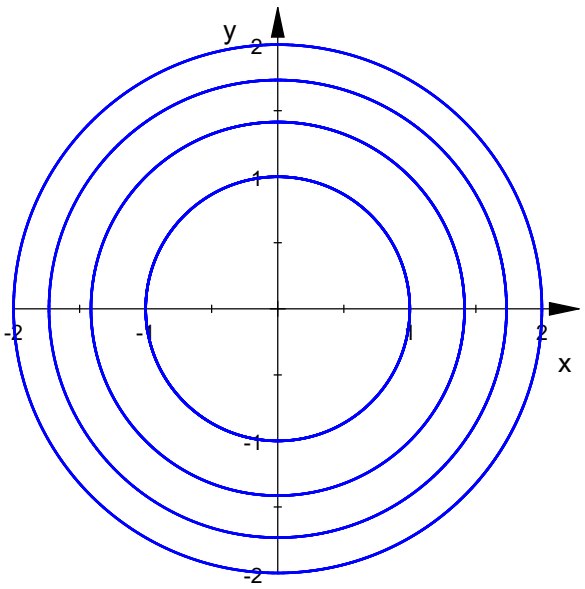
Man sieht hier auch noch besser, dass sich eine Parabel gedreht hat.
Nicht vergessen: Im 3D-Viewer ansehen!!!!!!!!!

Für die Höhenlinien allein, wie auf der Wanderkarte, wählt man z=konstant .

```
kreise:=(f(x,y)-k=0  $ k=0..4)
```

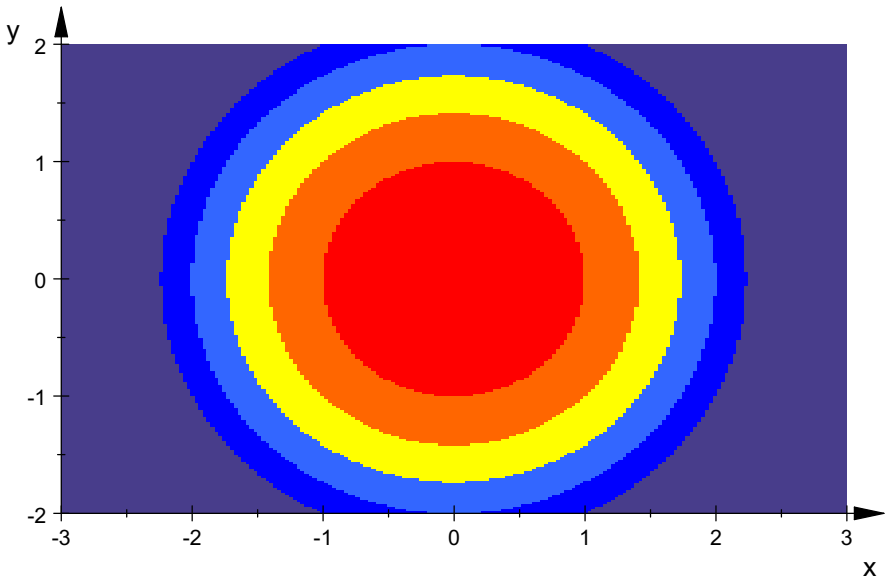
$$-x^2-y^2+4=0,\ -x^2-y^2+3=0,\ -x^2-y^2+2=0,\ -x^2-y^2+1=0,\ -x^2-y^2=0$$

```
hoehenlini:=plot::Implicit2d(kreise[j],x=-2..2,y=-2..2 )$j=1..4:
plot(hoehenlini, Scaling=Constrained)
```



Als Alternative für die Höhenlinien allein gibt es ein Extrawerkzeugin allen CAS.
es ist etwas aufwendiger

```
mycolor := proc(x, y, f)
begin
   if   f <= 1 then RGB::Red
   elif f <= 2 then RGB::Orange;
   elif f <=  3  then RGB::Yellow;
   elif f <=4 then RGB::BlueLight;
   elif f <= 5 then RGB::Blue;
   else RGB::SlateBlueDark;
   end_if;
end_proc:
plot(plot::Density((x^2+y^2), x = -3..3, y = -2..2,
                   Mesh = [60, 40],
                   FillColorFunction = mycolor,Mesh=[200,200]),
     Axes = Frame):
```

LLLLLLLLLL-------Level 3-------LLLLLLLLLL

Nun soll im Punkt P(1,1,z) eine Ebene gezeichnet werden.

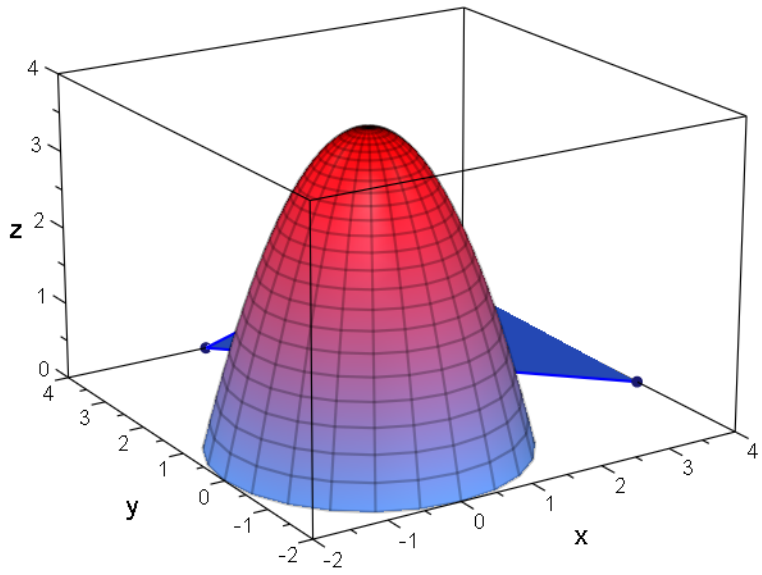
```
f(1,1)
```

2

Die z-Koodinate von P ist 2.

Verändere a so, dass die Ebene eine Tangentialebene zu sehen ist.

```
a:=4:
p1:=[a,0,0]: p2:=[0,a,0]:p3:=[1,1,2]:
p1g := plot::Point3d(p1):
p2g := plot::Point3d(p2):
p3g := plot::Point3d(p3):
eb1 := plot::Polygon3d([p1,p2,p3],Filled=TRUE):
plot(rund,eb1,p1g,p2g,p3g);
```



LLLLLLLLLL-------Level 4-------LLLLLLLLLL

Nun soll im Punkt P(1,1,2) eine Tangentialebene bestimmt werden.

Nun soll im Punkt P(1,1,2) eine Tangentialebene bestimmt werden.

Die Steigung nimmt man aus der ursprünglichen Parabel.
Dann ist ein wenig Geometrie im Raum fällig.
Besser sind die Methoden der 3D-Analysis, da hilft der Gradient (siehe Level 5).

```
diff(4-x^2,x)
```

$$-2 \cdot x$$

```
subs(%,x=sqrt(2))
```

$$-2 \cdot \sqrt{2}$$

```
ebene:=2*x+2*y+z=d:
subs(ebene,x=1 ,y=1, z=2)
```

$$6 = d$$

```
d:=op(%,1)
```

$$6$$

```
ebene
```

$$2 \cdot x + 2 \cdot y + z = 6$$

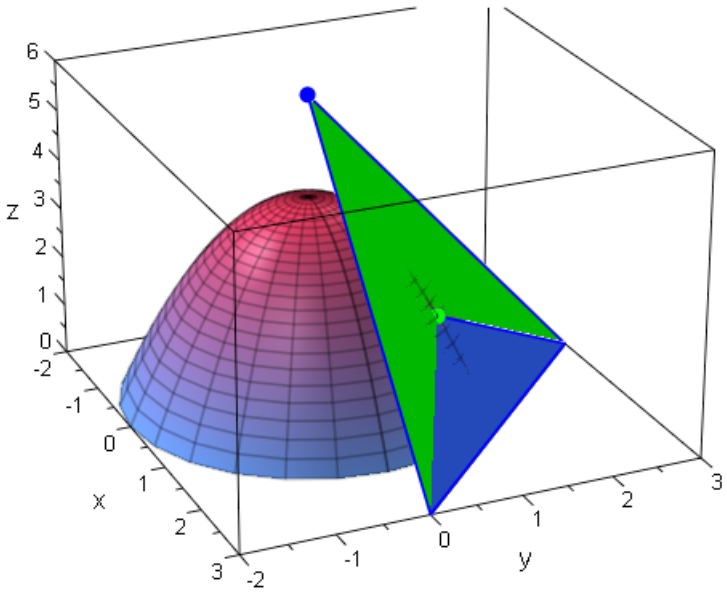
Die Achsen-Durchstoßpunkte sind leicht auszurechen:

```
p1:=[a,0,0]: p2:=[0,a,0]
```

$$[0,\ 3,\ 0]$$

a:=3: p4:=[0,0,6]:

```
p3g := plot::Point3d(p3, Color = RGB::Green, PointSize = 80):
p4g := plot::Point3d(p4, Color = RGB::Blue, PointSize = 50):
eb1 := plot::Polygon3d([p1,p2,p3],Filled=TRUE):
spur := plot::Polygon3d([p1,p2,p4], Closed=TRUE,Filled=TRUE, FillColor=[0,1,0]):
plot(rund,eb1,p4g,p3g, spur);
```



LLLLLLLLL-------Level 5 -------LLLLLLLLL

Nun soll im Punkt P(1,1,2) eine Tangentialebene mit den Methoden der

3D-Analysis bestimmt werden.

Partielle Ableitungen

```
fx:=diff(f(x,y),x);fy:=diff(f(x,y),y);
```

```
fx:=diff(f(x,y),x);fy:=diff(f(x,y),y);
```

$$-2 \cdot x$$

$$-2 \cdot y$$

```
subs(fx,x=1);subs(fy,y=1);
```

$$-2$$

$$-2$$

Die Tangentialebene in P0=[x0,y0,z0] ist

z-z0 = fx (x-x0) + fy (y-y0)

also:

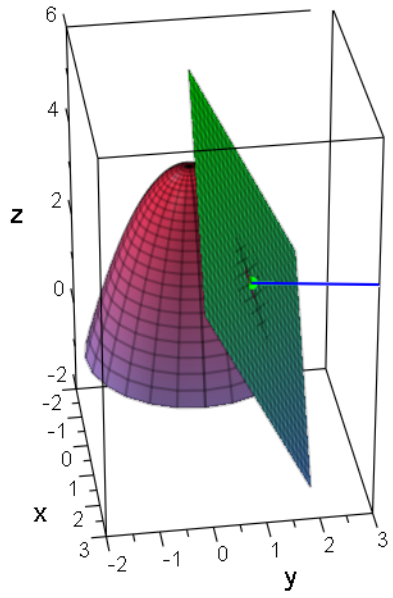
```
z-2=-2*(x-1)-2*(y-1)
```

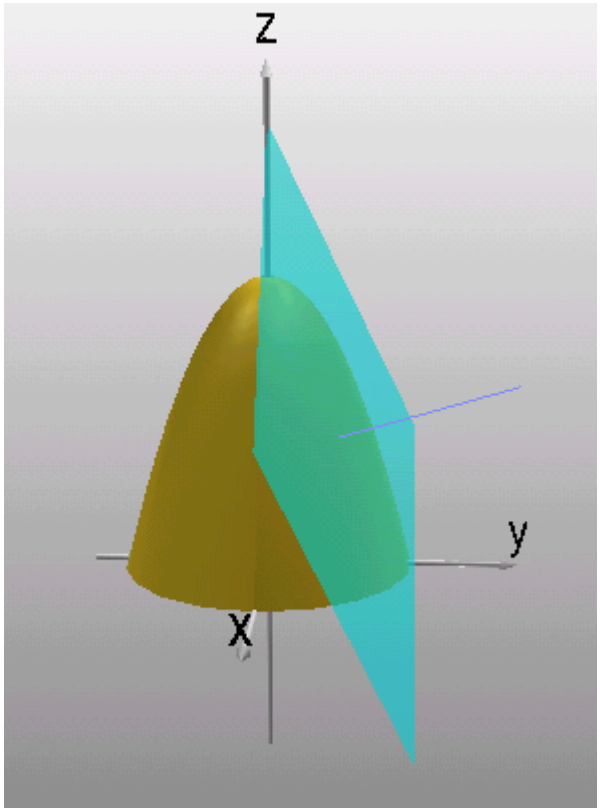
$$z-2=4-2 \cdot y-2 \cdot x$$

```
%+2
```

$$z=6-2 \cdot y-2 \cdot x$$

```
//[1,1,2]+[2,2,1]
tang:=plot::Function3d(6-2*y-2*x,x=0..2,y=0..2,FillColor=[0,1,0]):
p3g := plot::Point3d(p3,Color = RGB::Green, PointSize = 90):
stange:=plot::Line3d([1, 1,2],[3, 3,3]):
plot(rund,tang,p3g,stange,Scaling=Constrained):
```





Diese Ansicht wird nicht mehr erzeugt in MuPAD 4