Blatt 12

Aufgabe 46

```
> restart;
> with(plots):
> with(VectorCalculus):
> BasisFormat(false):
> f := (x, y) -> 4*x^2 - 3*x*y;
```

$$f := (x, y) \mapsto 4\,x^2 + (-(3\,x\,y)) \quad (1.1)$$

```
> param := t -> ( cos(t), sin(t) );
```

$$param := t \mapsto (\cos(t), \sin(t)) \quad (1.2)$$

```
> g := (f@param)(t);  # f eingeschränkt auf den Einheitskreis
```

$$g := 4\cos(t)^2 - 3\cos(t)\sin(t) \quad (1.3)$$

```
> dg := diff(g, t);   # Ableiten
```

$$dg := -8\cos(t)\sin(t) + 3\sin(t)^2 - 3\cos(t)^2 \quad (1.4)$$

```
> d2g := diff(dg, t);
> # Test
> d2g - diff(g, t$2);
```

$$d2g := 8\sin(t)^2 - 8\cos(t)^2 + 12\cos(t)\sin(t)$$

$$0 \quad (1.5)$$

```
> krit := solve({ dg = 0 }, t);
> # Die Lösungen werden im Interval [ -pi, pi ] gesucht. Test
> for kk from 1 to nops([ krit ]) do
    'dg'(rhs(krit[kk][1])) = simplify(subs(krit[kk], dg));
  od;
```

$$krit := \{t = \arctan(3)\}, \{t = \arctan(3) - \pi\}, \left\{t = -\arctan\left(\frac{1}{3}\right)\right\}, \left\{t = -\arctan\left(\frac{1}{3}\right) + \pi\right\}$$

$$dg(\arctan(3)) = 0$$

$$dg(\arctan(3) - \pi) = 0$$

$$dg\left(-\arctan\left(\frac{1}{3}\right)\right) = 0$$

$$dg\left(-\arctan\left(\frac{1}{3}\right) + \pi\right) = 0 \quad (1.6)$$

```
> # Werte der 2. Ableitung (hinreichendes Kriterium 2. Ordnung)
> d2g_krit := seq(simplify(subs(krit[kk], d2g)), kk = 1..nops([
  krit ]));
```

$$d2g\_krit := 10, 10, -10, -10 \quad (1.7)$$

```
> # Auswerten
```

```
> minMax[-1] := "Maximum":
> minMax[1] := "Minimum":
> minMax[0] := "???": # Bei 2. Ableitung = 0 müssen weitere
  Kriterien hinzugenommen werden
> for kk from 1 to nops([ krit ]) do
    p    := rhs(krit[kk][1]);
    gp   := subs(krit[kk], g);
    typ  := minMax[sign(d2g_krit[kk])];
    print(t = p, 'g'('t') = gp, d2g_krit[kk], typ);
  end do:
```

$$t = \arctan(3),\ g(t) = -\frac{1}{2},\ 10,\ \text{"Minimum"}$$

$$t = \arctan(3) - \pi,\ g(t) = -\frac{1}{2},\ 10,\ \text{"Minimum"}$$

$$t = -\arctan\left(\frac{1}{3}\right),\ g(t) = \frac{9}{2},\ -10,\ \text{"Maximum"}$$

$$t = -\arctan\left(\frac{1}{3}\right) + \pi,\ g(t) = \frac{9}{2},\ -10,\ \text{"Maximum"} \qquad (1.8)$$

```
> # Schön darstellen
> p_f := plot3d(f(x, y), x = -1.1 .. 1.1, y = -1.1 .. 1.1);
```

$$p\_f := PLOT3D(...) \qquad (1.9)$$

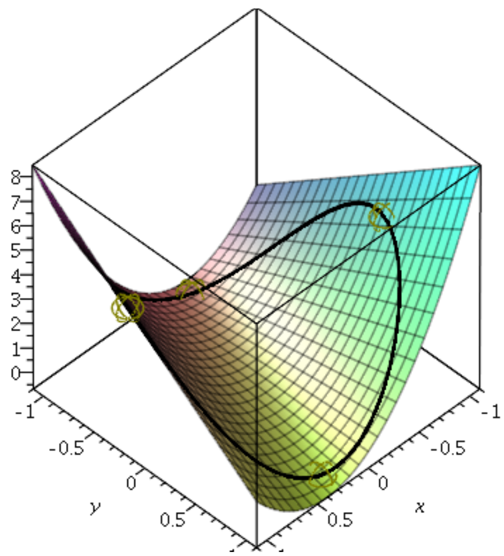
```
> p_g := spacecurve([ param(t)[1], param(t)[2], g ], t = -Pi ..
  Pi, color = black, thickness = 3);
```

$$p\_g := PLOT3D(...) \qquad (1.10)$$

```
> # Kritische Punkte, eingesetzt in die Parametrisierung
> kp3d := [ seq(subs(krit[kk], [ cos(t), sin(t), g ]), kk = 1..
  nops([ krit ])) ]:
> p_p := pointplot3d(kp3d, symbol=circle,  symbolsize = 50, color
  = yellow) ;
```

$$p\_p := PLOT3D(...) \qquad (1.11)$$

```
> display([ p_f, p_g, p_p ]);
```



Aufgabe 47

```
> restart;
> with(LinearAlgebra):
> with(ArrayTools):
> with(VectorCalculus):
> BasisFormat(false):
> SetCoordinates('cartesian'[x, y]);
```

$$cartesian_{x, y} \tag{2.1}$$

```
> f := (3*x^2 + x + y - 3*y^2) * exp(-(x^2 + y^2));
```

$$f := \left(3\,x^2 - 3\,y^2 + x + y\right)\,\mathrm{e}^{-x^2 - y^2} \tag{2.2}$$

```
> x0 := [ 0, 0 ];
> dx := [ x - x0[1], y - x0[2] ];
```

$$x0 := [0, 0]$$

$$dx := [x, y] \tag{2.3}$$

Manuelle Methode

(von Hand -- auf Papier! -- alle Summanden aufstellen)

```
> Tf[3] := simplify(
    subs([ x = x0[1], y = x0[2] ], f) +
    subs([ x = x0[1], y = x0[2] ], diff(f, x)) * dx[1] + subs([
 x = x0[1], y = x0[2] ], diff(f, y)) * dx[2] +
    (1/2)*(
     subs([ x = x0[1], y = x0[2] ], diff(f, x, y)) * dx[1] * dx
 [2] +
     subs([ x = x0[1], y = x0[2] ], diff(f, y, x)) * dx[2] * dx
 [1] +
     subs([ x = x0[1], y = x0[2] ], diff(f, x$2)) * dx[1]^2 +
     subs([ x = x0[1], y = x0[2] ], diff(f, y$2)) * dx[2]^2
    ) +
    (1/6)*(
     3 * subs([ x = x0[1], y = x0[2] ], diff(f, x$2, y)) * dx
 [1]^2 * y +
     3 * subs([ x = x0[1], y = x0[2] ], diff(f, y$2, x)) * dx
 [2]^2 * x +
     subs([ x = x0[1], y = x0[2] ], diff(f, x$3)) * dx[1]^3+
     subs([ x = x0[1], y = x0[2] ], diff(f, y$3)) * dx[2]^3));
```

$$Tf_3 := -x^3 - x^2\,y - y^2\,x - y^3 + 3\,x^2 - 3\,y^2 + x + y \tag{2.4}$$

Direkte Methode

(nach der Difinition)

```
> Gf := Gradient(f);
> Hf := Hessian(f);
> # Aber was ist mit der 3. Ableitung?
```

$$Gf := \begin{bmatrix} (6\,x+1)\,\mathrm{e}^{-x^2-y^2} - 2\,(3\,x^2-3\,y^2+x+y)\,x\,\mathrm{e}^{-x^2-y^2} \\ (-6\,y+1)\,\mathrm{e}^{-x^2-y^2} - 2\,(3\,x^2-3\,y^2+x+y)\,y\,\mathrm{e}^{-x^2-y^2} \end{bmatrix}$$

$$\begin{aligned} Hf := \Big[\big[&6\,\mathrm{e}^{-x^2-y^2} - 4\,(6\,x+1)\,x\,\mathrm{e}^{-x^2-y^2} - 2\,(3\,x^2-3\,y^2+x+y)\,\mathrm{e}^{-x^2-y^2} \\ &+4\,(3\,x^2-3\,y^2+x+y)\,x^2\,\mathrm{e}^{-x^2-y^2}, -2\,(6\,x+1)\,y\,\mathrm{e}^{-x^2-y^2} - 2\,(-6\,y \\ &+1)\,x\,\mathrm{e}^{-x^2-y^2} + 4\,(3\,x^2-3\,y^2+x+y)\,x\,y\,\mathrm{e}^{-x^2-y^2}\big], \\ \big[&-2\,(6\,x+1)\,y\,\mathrm{e}^{-x^2-y^2} - 2\,(-6\,y+1)\,x\,\mathrm{e}^{-x^2-y^2} + 4\,(3\,x^2-3\,y^2+x \\ &+y)\,x\,y\,\mathrm{e}^{-x^2-y^2}, -6\,\mathrm{e}^{-x^2-y^2} - 4\,(-6\,y+1)\,y\,\mathrm{e}^{-x^2-y^2} - 2\,(3\,x^2-3\,y^2+x \\ &+y)\,\mathrm{e}^{-x^2-y^2} + 4\,(3\,x^2-3\,y^2+x+y)\,y^2\,\mathrm{e}^{-x^2-y^2}\big]\Big] \end{aligned} \tag{2.5}$$

```
> # Alternativ (direkt mit der Definition höherer Ableitungen
  # mehrerer Veränderlicher):
> xy[1] := x; xy[2] := y;
```

$$xy_1 := x$$
$$xy_2 := y \tag{2.6}$$

```
> df := Array(1 .. 2, i -> diff(f, xy[i]));
> # Test
> 'df' - 'Gf' = convert(df, 'Vector') - Transpose(Gf);
```

$$df := \left[(6\,x+1)\,e^{-x^2-y^2} - 2\,(3\,x^2-3\,y^2+x+y)\,x\,e^{-x^2-y^2}, (-6\,y+1)\,e^{-x^2-y^2} - 2\,(3\,x^2-3\,y^2+x+y)\,y\,e^{-x^2-y^2}\right]$$

$$df - Gf = \begin{bmatrix} 0 & 0 \end{bmatrix} \tag{2.7}$$

```
> d2f := Array(1 .. 2, 1 .. 2, (i, j) -> diff(f, [ xy[i], xy[j] ]
  ));
> # Test
> 'd2f' - 'Hf' = convert(d2f, Matrix) - Hf;
```

$$d2f := [[6\,e^{-x^2-y^2} - 4\,(6\,x+1)\,x\,e^{-x^2-y^2} - 2\,(3\,x^2-3\,y^2+x+y)\,e^{-x^2-y^2} + 4\,(3\,x^2-3\,y^2+x+y)\,x^2\,e^{-x^2-y^2}, -2\,(6\,x+1)\,y\,e^{-x^2-y^2} - 2\,(-6\,y+1)\,x\,e^{-x^2-y^2} + 4\,(3\,x^2-3\,y^2+x+y)\,x\,y\,e^{-x^2-y^2}],$$
$$[-2\,(6\,x+1)\,y\,e^{-x^2-y^2} - 2\,(-6\,y+1)\,x\,e^{-x^2-y^2} + 4\,(3\,x^2-3\,y^2+x+y)\,x\,y\,e^{-x^2-y^2}, -6\,e^{-x^2-y^2} - 4\,(-6\,y+1)\,y\,e^{-x^2-y^2} - 2\,(3\,x^2-3\,y^2+x+y)\,e^{-x^2-y^2} + 4\,(3\,x^2-3\,y^2+x+y)\,y^2\,e^{-x^2-y^2}]]$$

$$d2f - Hf = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.8}$$

```
> # Die dritte Ableitung ist ein 3-Tensor!
> d3f := Array(1 .. 2, 1 .. 2, 1 .. 2, (i, j, k) -> diff(f, [ xy
  [i], xy[j], xy[k] ]));
```

$$d3f := \begin{bmatrix} \textit{1..2 x 1..2 x 1..2 Array} \\ \textit{Data Type: anything} \\ \textit{Storage: rectangular} \\ \textit{Order: Fortran\_order} \end{bmatrix} \tag{2.9}$$

```
> # So sieht die erste Zeile der ersten Ebene der 3. Ableitung
  aus:
> d3f(1, 1..2, 1);
```

$$[-36\,x\,e^{-x^2-y^2} - 6\,(6\,x+1)\,e^{-x^2-y^2} + 12\,(6\,x+1)\,x^2\,e^{-x^2-y^2} + 12\,(3\,x^2-3\,y^2+x+y)\,x\,e^{-x^2-y^2} - 8\,(3\,x^2-3\,y^2+x+y)\,x^3\,e^{-x^2-y^2}, -12\,y\,e^{-x^2-y^2} + 8\,(6\,x+1)\,x\,y\,e^{-x^2-y^2} - 2\,(-6\,y+1)\,e^{-x^2-y^2} + 4\,(3\,x^2-3\,y^2+x+y)\,y\,e^{-x^2-y^2} + 4\,(-6\,y+1)\,x^2\,e^{-x^2-y^2} - 8\,(3\,x^2-3\,y^2+x+y)\,x^2\,y\,e^{-x^2-y^2}] \tag{2.10}$$

```
> # Taylorpolynom:
  # T_k[f,x_0](x) = sum_i=0^k D^i f(x_0) [(x - x_0), ..., (x -
  x_0)]/i!
> # Auswerten der Ableitungen
> auswerten := (f, v) -> subs([ x = v[1], y = v[2] ], f);
```

(2.11)

$$auswerten := (f, v) \mapsto subs([x = v_1, y = v_2], f) \quad \mathbf{(2.11)}$$

```
> # Prozedur um einen k-Tensor auf k Vektoren anzuwenden,
  # die als Array übergeben werden
> anwenden := proc (f, vs)
   description "Wende einen k-Tensor auf k Vektoren an.";
   local k, ind, ii, kk, curSumd, curSum;

   # Anzahl Vektoren/Stufe des Tensors
    k  := nops(vs);

   # alle möglichen Index-k-Tupel des Tensors
    ind := indices(f);

   curSum := 0;
   for ii in ind do
     # f[ii[1]][ii[2]]...[ii[k]] auswerten
     curSumd := f(seq(ii[ll], ll = 1..nops(ii)));

     for kk from 1 to nops(vs) do
       # ii-ter Eintrag des kk-ten Vektors
         curSumd := curSumd * vs[kk][ii[kk]];
     end do;
     curSum := curSum + curSumd;
   end do;
   return curSum;
  end proc:
> #f(seq(ii[ll], ll = 1..nops(ii))) * product(dx[ii[i]], i = 1..
  nops(ii));
> # Summanden der Taylorentwicklung (nur zum Ansehen):
> 'f'('x0') = auswerten(f, x0);
> 'df'('x0') * ('x' - 'x0') = anwenden(auswerten(df, x0), [ dx ]
  ) ;
> ('x' - 'x0')^'T' * 'd2f'('x0') * ('x' - 'x0') = anwenden
  (auswerten(d2f, x0), [ dx, dx ]);
> 'd3f'('x0')['x' - 'x0','x' - 'x0','x' - 'x0'] = anwenden
  (auswerten(d3f, x0), [ dx, dx, dx ]);
```

$$f(x0) = 0$$

$$df(x0)\ (x - x0) = e^0\, x + e^0\, y$$

$$(x - x0)^T\, d2f(x0)\ (x - x0) = 6\, e^0\, x^2 - 6\, e^0\, y^2$$

$$d3f(x0)_{x - x0,\, x - x0,\, x - x0} = -6\, e^0\, x^2\, y - 6\, e^0\, y^2\, x - 6\, e^0\, x^3 - 6\, e^0\, y^3 \quad \mathbf{(2.12)}$$

```
> # Jetzt zusammensetzen
> Tf_alt[3] := auswerten(f, x0)/factorial(0) + anwenden(auswerten
  (df, x0), [ dx ])/factorial(1)
        + anwenden(auswerten(d2f, x0) , [ dx, dx ])/factorial(2)
        + anwenden(auswerten(d3f, x0) , [ dx, dx, dx ])/factorial
  (3);
```

```
> Tf[3] := simplify(Tf[3]);
```

$$Tf\_alt_3 := e^0\, x + e^0\, y + 3\, e^0\, x^2 - 3\, e^0\, y^2 - e^0\, x^2\, y - e^0\, y^2\, x - e^0\, x^3 - e^0\, y^3$$

$$Tf_3 := -x^3 - x^2\, y - y^2\, x - y^3 + 3\, x^2 - 3\, y^2 + x + y \qquad \textbf{(2.13)}$$

Auf die 1D-Version zurückführen

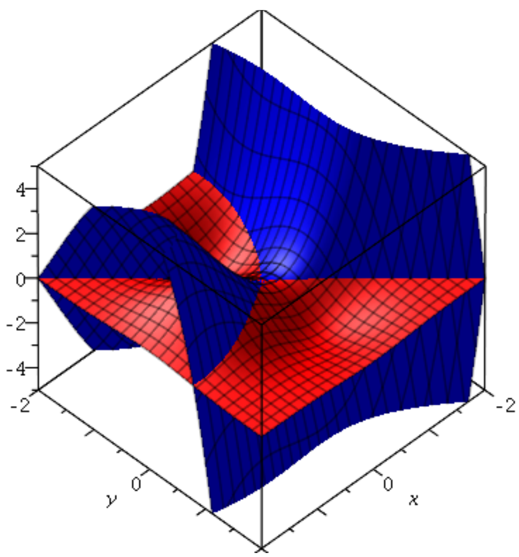
(schreibe g[x_0,x](t) := f(x_0 + t*(x - x_0) und bestimme die Taylor-Entwicklung von g. Dann ergbit sich T_k(f, x_0)(x) = T_k(g[x_0,x], 0)(1).)

```
> # Darf jemand anderes machen ;-).
> # Probe
> Tf[3] - Tf_alt[3];
```

$$0 \qquad \textbf{(2.14)}$$

Plotten

```
> plot3d([ f, Tf[3] ], x=-2..2, y=-2..2, view=-5..5,
       color = [ 'red', 'blue' ]);
```




```
> #plot3d([ f, Tf[3] ], x=-1..1, y=-1..1, color = [ 'red', 'blue'
  ]);
```

Aufgabe 48

```
> # Satz über die Implizite Funktion
> restart:
> with(LinearAlgebra):
> f := (x, y) -> exp(y) + y^3 + x^3 + x^2 - 1;
```

$$f := (x, y) \mapsto e^{y} + y^{3} + x^{3} + x^{2} - 1 \tag{3.1}$$

```
> implFun := proc(f, x)
    local Dyf, r, dim, g;
   # Bestimme den Rang des y-Anteils der Jacobi-Matrix
    Dyf := diff(f(x, y), y);
    dim := max(Dimension(convert([f(x,y)], Matrix)));
     r  := Rank(convert([Dyf], Matrix));
   printf("Dimension: %d, Rang: %d\n", dim, r);
    # Die implizite Funktion g, die als Lösung von f(x, g(x)) = 0
  gegeben
    # ist, gibt es, wenn der Rang voll ist.
    if r = dim then
      print("y-Anteil der Jacobi-Matrix ist invertierbar,
  implizite Funktion existiert.");
       g := rhs(solve({ f(x, g) = 0 }, { g })[1]);
    else
       print("y-Anteil der Jacobi-Matrix ist nicht invertierbar,
  implizite Funktion existiert NICHT.");
     end if;
    return g;
  end proc;
```

implFun := **proc**(*f*, *x*) **(3.2)**

local *Dyf*, *r*, *dim*, *g*;

Dyf := *diff*(*f*(*x*, *y*), *y*);

dim := max(*LinearAlgebra*:-*Dimension*(*convert*([*f*(*x*, *y*)], *Matrix*)));

r := *LinearAlgebra*:-*Rank*(*convert*([*Dyf*], *Matrix*));

printf("Dimension: %d, Rang: %d\n", *dim*, *r*);

if *r* = *dim* **then**

print("y-Anteil der Jacobi-Matrix ist invertierbar, implizite Funktion existiert.");

g := *rhs*(*solve*({*f*(*x*, *g*) = 0}, {*g*})[1])

else

print("y-Anteil der Jacobi-Matrix ist nicht invertierbar, implizite Funktion existiert NICHT.")

end if;

return *g*

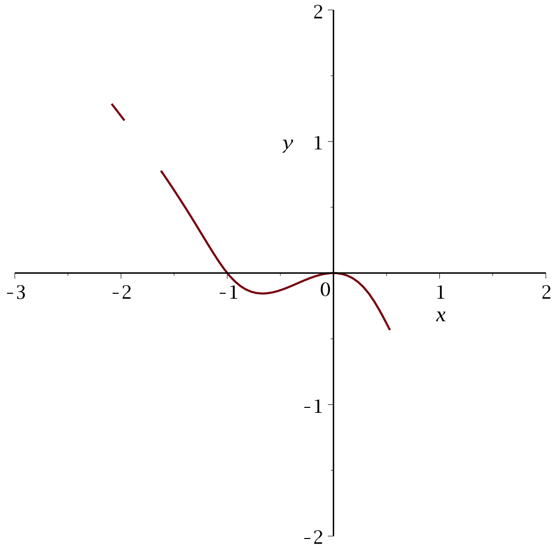
end proc

```
> g := implFun(f, x);
Dimension: 1, Rang: 1
```

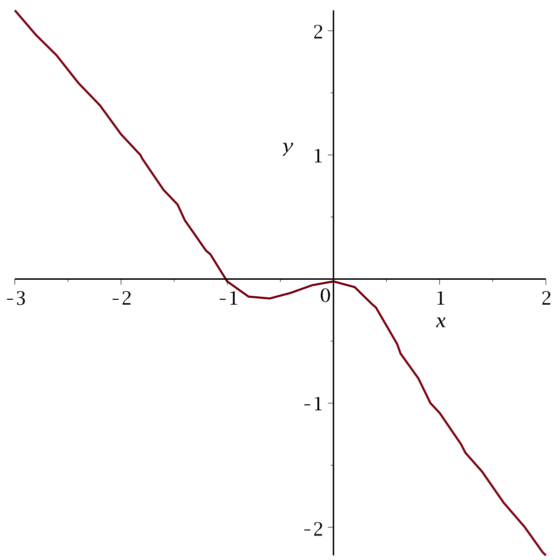
"y-Anteil der Jacobi-Matrix ist invertierbar, implizite Funktion existiert."

$$g := RootOf\left(\mathrm{e}^{-\_Z} + \_Z^3 + x^3 + x^2 - 1\right) \tag{3.3}$$

```
> plot(g, x = -3..2, y = -2..2, numpoints = 100);
> # kein sehr schöner Plot ...
```




```
> # Alternative
> with(plots):
> implicitplot({ f(x, y) = 0 }, x = -3..2, y=-5..5) ;
```



Aufgabe 49

```
> restart:
> with(LinearAlgebra):
> with(VectorCalculus):
> with(plots):
> BasisFormat(false):
> f := x^2 * y;
```

$$f := x^2 y \quad \textbf{(4.1)}$$

```
> g := x^2 + y^2 - 3;
```

$$g := x^2 + y^2 - 3 \quad \textbf{(4.2)}$$

```
> L := f + lambda * g;
```

$$L := x^2 y + \lambda (x^2 + y^2 - 3) \quad \textbf{(4.3)}$$

```
> dL := [ diff(L, x), diff(L, y), diff(L, lambda) ];
```

$$dL := [2 \lambda x + 2 x y, 2 \lambda y + x^2, x^2 + y^2 - 3] \quad \textbf{(4.4)}$$

```
> krit := seq(allvalues(s), s = solve({ dL[1] = 0, dL[2] = 0, dL
```

```
[3] = 0 }, { x, y, lambda }));
```

$$krit := \{\lambda = 0, x = 0, y = \sqrt{3}\}, \{\lambda = 0, x = 0, y = -\sqrt{3}\}, \{\lambda = -1, x = \sqrt{2}, y = 1\}, \{\lambda = -1, x = -\sqrt{2}, y = 1\}, \{\lambda = 1, x = \sqrt{2}, y = -1\}, \{\lambda = 1, x = -\sqrt{2}, y = -1\} \tag{4.5}$$

```
> HL := Hessian(L, [ x, y ]);
```

$$HL := \begin{bmatrix} 2y + 2\lambda & 2x \\ 2x & 2\lambda \end{bmatrix} \tag{4.6}$$

```
> GradG := Gradient(g, [ x, y ]);
```

$$GradG := \begin{bmatrix} 2x \\ 2y \end{bmatrix} \tag{4.7}$$

```
> # Wenn die letzte Spalte der Ausgabe positiv für alle Werte
  # von s[1], s[2] ist, so haben wir ein lokales Minimum; wenn
  # negativ, dann ein Maxiumum, wenn dem nicht so ist, haben
  # wir kein sicheres Kriterium.
> for kk from 1 to nops([ krit ]) do
    p   := subs(krit[kk], [ x, y ]);
    fp  := subs(krit[kk], f);
   typ := minMax[sign(d2g_krit[kk])];
   HLp := subs(krit[kk], HL);

   # Tangentialvektor bestimmen. Der Gradient steht
   # orthogonal auf dem Rand des Gebietes, d.h.
   # Tangentialen stehen orthogonal auf dem Gradienten.
    GradGp := subs(krit[kk], GradG);
   print("orthogonale:", GradGp);
    tans := [ solve({ GradGp[1] * s[1] + GradGp[2] * s[2] = 0 },
           { s[1], s[2] }) ];
   print("Tangentialraum:", tans);
    v := subs(tans[1], <s[1], s[2]>);
   kriterium := seq(Transpose(v).HLp.v, s in tans);
    print([x,y] = p, 'f'(['x','y']) = fp, kriterium);
  end do:
```

$$\text{"orthogonale:"}, \begin{bmatrix} 0 \\ 2\sqrt{3} \end{bmatrix}$$

$$\text{"Tangentialraum:"}, [\{s_1 = s_1, s_2 = 0\}]$$

$$[x, y] = [0, \sqrt{3}], f([x, y]) = 0, 2 s_1^2 \sqrt{3}$$

$$\text{"orthogonale:"}, \begin{bmatrix} 0 \\ -2\sqrt{3} \end{bmatrix}$$

$$\text{"Tangentialraum:"}, \left[\{s_1 = s_1, s_2 = 0\}\right]$$

$$[x, y] = \left[0, -\sqrt{3}\right], f([x, y]) = 0, -2\, s_1^2 \sqrt{3}$$

$$\text{"orthogonale:"}, \begin{bmatrix} 2\sqrt{2} \\ 2 \end{bmatrix}$$

$$\text{"Tangentialraum:"}, \left[\left\{s_1 = s_1, s_2 = -\sqrt{2}\, s_1\right\}\right]$$

$$[x, y] = \left[\sqrt{2}, 1\right], f([x, y]) = 2, -12\, s_1^2$$

$$\text{"orthogonale:"}, \begin{bmatrix} -2\sqrt{2} \\ 2 \end{bmatrix}$$

$$\text{"Tangentialraum:"}, \left[\left\{s_1 = s_1, s_2 = \sqrt{2}\, s_1\right\}\right]$$

$$[x, y] = \left[-\sqrt{2}, 1\right], f([x, y]) = 2, -12\, s_1^2$$

$$\text{"orthogonale:"}, \begin{bmatrix} 2\sqrt{2} \\ -2 \end{bmatrix}$$

$$\text{"Tangentialraum:"}, \left[\left\{s_1 = s_1, s_2 = \sqrt{2}\, s_1\right\}\right]$$

$$[x, y] = \left[\sqrt{2}, -1\right], f([x, y]) = -2, 12\, s_1^2$$

$$\text{"orthogonale:"}, \begin{bmatrix} -2\sqrt{2} \\ -2 \end{bmatrix}$$

$$\text{"Tangentialraum:"}, \left[\left\{s_1 = s_1, s_2 = -\sqrt{2}\, s_1\right\}\right]$$

$$[x, y] = \left[-\sqrt{2}, -1\right], f([x, y]) = -2, 12\, s_1^2 \qquad \textbf{(4.8)}$$

```
> # Jetzt noch plotten (optional)
> p_f := plot3d(f, x = -sqrt(3)*1.1..sqrt(3)*1.1, y = -sqrt(3)*
  1.1..sqrt(3)*1.1, view = -3..3);
```

$$p\_f := PLOT3D(...) \qquad \textbf{(4.9)}$$

```
> param := [ cos(t)*sqrt(3), sin(t)*sqrt(3) ];
> p_t := spacecurve([ param[1], param[2], subs([ x = param[1], y
  = param[2] ], f) ], t = -Pi..Pi, color = black, thickness = 3);
```

$$param := \left[\cos(t)\sqrt{3}, \sin(t)\sqrt{3}\right]$$

$$p\_t := PLOT3D(...) \qquad \textbf{(4.10)}$$

```
> # Kritische Punkte
> kp3d := [ seq(subs(krit[kk], [ x, y, f ]), kk = 1..nops([ krit
  ])) ];
> p_p := pointplot3d(kp3d, symbol=circle, symbolsize = 50, color
  = yellow);
```

$kp3d := [[0, \sqrt{3}, 0], [0, -\sqrt{3}, 0], [\sqrt{2}, 1, 2], [-\sqrt{2}, 1, 2], [\sqrt{2}, -1, -2], [-\sqrt{2}, -1, -2]]$

$$p\_p := PLOT3D(...) \quad (4.11)$$

```
> display([ p_f, p_t, p_p ]);
```