# Blatt 11

## Aufgabe 42

```
> restart;
> with(LinearAlgebra):
> f := (x, y, z) -> x^2 - y^2 + z^2 - (x^2 + 2*y^2 +4*z^2)^2;
> Df := [ diff(f(x, y, z), x), diff(f(x, y, z), y), diff(f(x, y,
  z), z) ];
> D2f := < < diff(f(x, y, z), x$2), diff(f(x, y, z), [ x, y ]),
  diff(f(x, y, z), [ x, z ]) > |
  < diff(f(x, y, z), [ y, x ]), diff(f(x, y, z), y$2), diff(f(x,
  y, z), [ y, z ]) > |
  < diff(f(x, y, z), [ z, x ]), diff(f(x, y, z), [ z, y ]), diff
  (f(x, y, z), z$2) > >;
> kritischePunkte := solve([ seq(Df[i] = 0, i = 1..3) ], [ x, y,
  z ]):
> kritischePunkte := seq(allvalues(kritischePunkte[kk]), kk = 1..
  nops(kritischePunkte));
> # Reelle Lösungen
> kritischePunkte := [ seq(kritischePunkte[kk], kk in [ 1, 2, 3,
  6, 7 ]) ];
> # Prüfe Definitheit der Hesse-Matrix
> seq(print(kritischePunkte[kk], 'EW' = Eigenvalues(subs
  (kritischePunkte[kk], D2f))), kk = 1..nops(kritischePunkte));
> # Das heißt: 0 und (0,0,+-sqrt(2)/8) sind Sattelpunkte,
  # die anderen beiden lokale Maxima
> plot3d(f(x, 0, y), x = -1..1, y = -1/2..1/2, style =
  patchcontour, contours=35, view = -0.3 .. 0.3, numpoints =
  3000);
> plot3d(f(x, y, 0), x = -1..1, y = -1/2..1/2, style =
  patchcontour, contours=35, view = -0.3 .. 0.3, numpoints =
  3000);
```

## Aufgabe 43

```
> restart;
> with(LinearAlgebra):
> with(VectorCalculus):
> with(plots):
> q := < x, y, z >;
> q1 := < 1, 0, 0 >;
> q2 := < 0, 1, 0 >;
> f := 3 / VectorNorm(q1 - q, 2) + 2 / VectorNorm(q2 - q, 2);
> Gradf := Gradient(f, [ x, y, z ]);
```

```
> fieldplot3d(Gradf, x = -0.5..1.5, y = -0.5..1.5, z = -1..1,
  thickness = 2, axes = boxed);
```

## Aufgabe 44

```
> restart;
> f := x -> -(x^3 + 3*x^2 + 4*x + 3) * exp(-x);
> df := D(f);
> d2f := D(df);
```

(a)

```
> plot([ f(x), df(x), d2f(x) ],
      x = -2..6, y = -5..5,
      color = [ blue, red, magenta ], legend = [ 'f', "f'", "f''"
  ] ,
     numpoints = 1000);
```

(b)

```
> minimize(f(x), x = -1..2);
> maximize(f(x), x = -1..2);
```

(c)

```
> kritischeStellen := solve({ df(x) = 0 }, { x });
> #evalf(kritischeStellen) ;
> rhs(kritischeStellen[1][1]);
> f(rhs(kritischeStellen[2][1]));
> mm[false] := "Maximum"; mm[true] := "Minimum";
> seq(print([ f(rhs(x[1])), d2f(rhs(x[1])), mm[is(d2f(rhs(x[1])),
  positive)] ]), x in [ kritischeStellen ]);
```

(d)

```
> tangente := x -> f(0) + (x - 0) * df(0);
> plot([ f(x), tangente(x) ], x = -2..2);
> schnittpunkte := solve({ tangente(x) = f(x) }, x);
> schnittpunkte := allvalues(schnittpunkte);
> evalf(schnittpunkte); # Hmmmm
> s1 := fsolve({ tangente(x) = f(x) }, x);
> s1 := rhs(s1[1]);
> s2 := fsolve({ tangente(x) = f(x) }, x, avoid = { x = s1 });
> s2 := rhs(s2[1]);
> s3 := fsolve({ tangente(x) = f(x) }, x, avoid = { x = s1, x =
  s2 });
> s3 := rhs(s3[1]);
```

## Aufgabe 45

```
> restart;
> with(VectorCalculus):
> SetCoordinates('cartesian'[x[1], x[2], x[3]]);
```

```
> BasisFormat(false);
> f := x -> exp(x[1] * x[2]) * arctan(x[2] * x[3]);
```
(a)
```
> df := x -> < diff(f(x), x[1]), diff(f(x), x[2]), diff(f(x), x
  [3]) >;
> gr := Gradient(f(x), [ x[1], x[2], x[3] ]);
> # Test
> VectorField(df(x)) - gr;
> d2f := x -> < < diff(f(x), x[1]$2),     diff(f(x), x[2], x[1]),
  diff(f(x), x[3], x[1]) > |
                < diff(f(x), x[1], x[2]), diff(f(x), x[2]$2),
  diff(f(x), x[3], x[2]) > |
                < diff(f(x), x[1], x[3]), diff(f(x), x[2], x[3]),
  diff(f(x), x[3]$2)     > >;
> H := Hessian(f(x), [ x[1], x[2], x[3] ]);
> # Test
> d2f(x) - H;
```
(b)
```
> r := (s, x) -> f(<x[1], x[2], x[3]> + <1, 1, 1> * s);
> ra := unapply(eval(diff(r(s, x), s), s = 0), x);
> # Alternative Definition
> ra_Grad := gr . <1,1,1>;
> # Test
> simplify(ra_Grad - ra(x));
```
(c)
```
> points := <0, 1, 0>, <1, 1, 1>;
> seq(print(ra(p)), p in points);
```