Computergestuetzte Mathematik zur Analysis

Lektion 11 (14. Januar)

Gradienten und Vektorfelder

```
> restart:
> with(VectorCalculus):
> BasisFormat(false);
> f := a*x^2 + b*y^2 + c*z^2;
> gr := Gradient(f, [x, y, z]);
> gr . <b*y, -a*x,0>;
> vf := VectorField(<b*y, -a*x, 0>, cartesian[x,y,z]);
> gr . vf; # Skalarprodukt
```

Zeichnungen von Vektorfeldern

```
> restart:
> with(VectorCalculus):
> BasisFormat(false);
> vf1 := VectorField(<-y, x>, cartesian[x,y]);
  vf2 := VectorField(<x, y>, cartesian[x,y]);
  vf3 := VectorField(<y, x>, cartesian[x,y]);
> with(plots):
> fieldplot(vf1, x = -1 .. 1, y = -1 .. 1, thickness = 2);
> fieldplot(vf2, x=-1..1,y=-1..1,thickness=2);
> fieldplot(vf3,x=-1..1,y=-1..1,thickness=2);
> with(LinearAlgebra):
> vf2 := vf2/Norm(vf2, 2);
> fieldplot(vf2, x = -1 .. 1, y = -1 .. 1, thickness = 2, color =
  sqrt(x^2 + y^2));
> k := -1/sqrt(x^2 + (y-1)^2 + 1) + 1/sqrt((x-1)^2 + (y+1)^2 + 1)
  + 1/sqrt((x+1)^2 + (y+1)^2 + 1);
> gr := Gradient(k, [x,y]);
> fieldplot(gr, x = -2 .. 2, y = -2.3 .. 2.3, axes = frame,
  thickness = 2);

> divgr := Divergence(gr):
> plot3d(divgr,x=-2..2 ,y=-2..2,lightmodel=none,color=divgr);
> gr3 := Gradient(k, [x,y,z]);
> f3 := fieldplot3d(gr3, x = -2 .. 2, y = -2.3 .. 2.3, z = -1 ..
  1, color = black, grid = [10, 10, 20]):
> f3;
> pl3 := plot3d(k, x = -2 .. 2, y = -2.3 .. 2.3, shading = zhue,
  style = patchcontour,lightmodel=none, numpoints = 3000):
> display({f3, pl3}, axes = frame, orientation = [-90,0]);
> k3 := 1/sqrt((x-1)^2 + y^2 + z^2 + 1) - 1/sqrt((x+1)^2 + y^2 +
  z^2 + 1);
> gr := Gradient(k3, [x,y,z]);
> fieldplot3d(gr, x = -1.5..1.5, y = -1.5..1.5, z = -1.5..1.5,
  orientation = [65, 30], axes = boxed, thickness = 2);
```

Jacobimatrix

```
> restart:
> with(VectorCalculus):
> BasisFormat(false):
> F := <F1(x,y,z), F2(x,y,z)>;
> Jacobian(F, [x,y,z]);
> F3 := <F[1], F[2], 0>;
> Jacobian(F3, [x,y,z]);
> with(LinearAlgebra):
> jac := SubMatrix(%%, 1..2, 1..3);
> F := <x^2 + 2*x + 2 + y^2 - 2*y, x^2 + 2*x - y^2 + 2*y, x*y - x
  + y -1>;
> Jacobian(F, [x,y,z]);
> J := SubMatrix(%, 1..3, 1..2);
> Rank(J); # Vorsicht falsch
> ReducedRowEchelonForm(J);
> J;
> J1 := RowOperation(J, [2,1], -1);
> J2 := RowOperation(J1, [3,2]);
> J3 := RowOperation(J2, 1, y-1);   # ausser fuer y = 1
> J4 := RowOperation(J3, 2, 2*x+2);   # ausser fuer x = -1;
> RowOperation(J4, [2,1], -1);
> J5 := map(expand, %);
> map(factor, J5);
```

Also ist fuer x <>-1 und y<>1 der Rang tatsaechlich 2. Wir testen den Extremfall

```
> subs(x = -1, y = 1, J);
```

Hessematrix

f : R^n --> R

```
> with(VectorCalculus):
> f := (x, y, z) -> exp(x^2+y^2+z);
> Hessian(f(x, y, z), [x, y, z]);
> g := exp(x^2+y^2+z);
> Hessian(g, [x, y, z]);
> with(LinearAlgebra):
> IsDefinite(subs([x = 1, y = 2, z = 1], ⁇));
```

Lokale Extrema

```
> restart;
> with(VectorCalculus): with(LinearAlgebra):
> f:= -1/2*x^4 - x^2*y^2-1/2*y^4+x^3-3*x*y^2;
> plot3d(f,x=-2..2,y=-2..2,view=-1..1,style=patchcontour);
> g:=Gradient(f,[x,y]);
> H:=Hessian(f,[x,y]);
> solve(convert(g,set),{x,y});
> L:=solve(convert(g,set),{x,y},Explicit,DropMultiplicity);

> HH := seq(subs(L[k],H),k=1..4);
> IsDefinite(HH[2]);
> IsDefinite(HH[2],query=negative_definite);
> IsDefinite(HH[3],query=negative_definite);
```

Noch ein Beispiel

```
> f := (x^2+y-11)^2 + (x+y^2-7)^2;
> plot3d(f,x=-5..5,y=-5..5); # Himmelblaufunktion
> BasisFormat(false):
> g:=Gradient(f,[x,y]);

> H:=map(factor,Hessian(f,[x,y]));
> _EnvAllSolutions := true;
> L:=solve([g[1]=0,g[2]=0],{x,y});
> AVL := seq(allvalues(L[k]),k=1..3);
> seq(simplify(evalc(AVL[k])) , k=1..7);
> seq(simplify(evalc(subs(AVL[k],g))) , k=1..7);
> L1:=L[1];
> L2_ := allvalues(L[2]);
> L2[1] := {simplify(evalc(L2_[1][1])), simplify(evalc(L2_[1][2])
  )};
> L2[2] := {simplify(evalc(L2_[2][1])), simplify(evalc(L2_[2][2])
  )};
> L2[3] := {simplify(evalc(L2_[2][1])), simplify(evalc(L2_[2][2])
  )};
> L3 := evalf(allvalues(L[3]));
> subs(L[1],H);
> IsDefinite(??);
> subs(L2[1],H);
> IsDefinite(??);
> subs(L2[2],H);
> IsDefinite(??);
> subs(L2[3],H);
> IsDefinite(??);
> seq(IsDefinite(subs(L3[k],H),query=negative_definite),k=1..5);
> seq(IsDefinite(subs(L3[k],H),query=positive_definite),k=1..5);
> seq(IsDefinite(subs(L3[k],H),query=positive_semidefinite),k=1.
  .5);
> seq(IsDefinite(subs(L3[k],H),query=negative_semidefinite),k=1.
  .5);
> seq(IsDefinite(subs(L3[k],H),query=indefinite),k=1..5);
```