Computergestuetzte Mathematik zur Analysis

Lektion 10 (19. Dezember)

```
> restart: with(plots):
```

Partielle Ableitungen

```
> f := exp(x);
> df := Diff(f, x);
> value(df);
> g := exp(a*x + b*y + c*z);
> dg := Diff(g, x);
> value(dg);
> d123g := Diff(g, x, y, y, z$3):
> value(d123g);
> h := (x, y, z) -> sin(a*x + b*y + c*z);
> D[2](h);
> D[1, 2, 2, 3$3](h);
> f := (x, y) -> sin(sqrt(x^2 + y^2)) * ((x-1/4)^2-(y-1/3)^2);
> p1 := plot3d(f-.05, -2 .. 2, -2 .. 2, style = surfacecontour,
  contours=30, shading = zhue):
> display(p1,orientation=[-40,50]);
> y_schnittkurve := [t, y, f(t, y), y = -2..2];
> tangente := f(1/2,-1) + D[2](f)(1/2,-1) + D[2](f)(1/2,-1)*y:
> plot([[y,f(1/2,y),y=-2..2],[y,tangente,y=-2..0]],color=
  [black,red], thickness = 3);
> y_schnitte :=spacecurve({seq(y_schnittkurve, t=-2..2,1/2)},
  color = black, thickness = 2);
> display([p1,y_schnitte],orientation=[-40,50]);
> p := <-3/2, -1, f(-3/2, -1)>;
> Dy := D[2](f)(-3/2, -1);
> y_tan := p + t.<0,1,Dy>;
> y_tan := simplify(y_tan);
> y_tan_pl := spacecurve(convert(y_tan, list), t = -1 .. 3/2,
  color = red, thickness = 3):
> display({p1,y_schnitte,y_tan_pl}, orientation=[-40,50]);
> grad := <D[1](f)(-3/2,-1),D[2](f)(-3/2,-1)>; ngrad := norm
```

```
(grad,2): dgrad:= simplify(grad/ngrad):
> grad_tan := p+t.<dgrad[1],dgrad[2],ngrad>;
> grad_tan := simplify(grad_tan);
> grad_tan_pl :=spacecurve(convert(grad_tan, list), t = -1 ..
  3/2, color = blue, thickness = 3):
> display({p1,y_schnitte,grad_tan_pl}, orientation=[90,00]);
```

Ableitungen von Vektorfunktionen

```
> restart:
> v := <t, t^2, t^3>;
> diff(v, t):
> with(VectorCalculus):
> diff(v, t);
> BasisFormat(false);
> dv := diff(v, t);
> with(plots):
> spacecurve(v, t = -3 .. 3,thickness=3);
```

Moebiusband

```
> restart: with(plots):
> M := <cos(t)*(1 + s*cos(t/2)), sin(t)*(1+s*cos(t/2)), s*sin
  (t/2)>;
  p1:= plot3d(M, t = 0 .. Pi, s=-1/2..0,color=blue);
  p2:= plot3d(M, t = 0 .. Pi, s=0..1/2,color=red);
  p3:= spacecurve(subs(s=1/2+0.02,convert(M,list)),t=0..Pi,
  color=coral,thickness=5);
  display({p1,p2,p3});

> Seele := subs(s = 0, M);
> with(VectorCalculus):
> BasisFormat(false);
> Mt := diff(Seele, t);
> Ms := diff(M, s);
> with(LinearAlgebra):
> Normale := CrossProduct(Ms, Mt);
> pl1 := plot3d(M, t = 0 .. 2*Pi, s = -1/3 .. 1/3, grid = [60,
  5], color = red):
> EinheitsNormale := simplify(Normale/Norm(Normale, 2))
  assuming t::real:
> EinheitsNormale[1];
> flaeche := convert(Seele + s*EinheitsNormale, list);
> pl2 := plot3d(flaeche, t = 0 .. 2*Pi, s = 0 .. .4, color = s,
```

```
   numpoints = 3000, style = patchnogrid):
> with(plots):
> display({pl1, pl2}, orientation = [-78, -159]);
```

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);
> 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, 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);
```

Divergenz und Rotation

```
> with(VectorCalculus):
> SetCoordinates(cartesian[x, y, z]):
> BasisFormat(false);
> F := VectorField(<x*y,-y*z, z*z>);
> Divergence(F);
> Divergence(Gradient(h(x, y, z)));
> Laplacian(h(x, y, z));
> E:= VectorField(<a(x,y,z),b(x,y,z),c(x,y,z)>);
> Curl(E); # Rotation  Gradient X E
> Curl(gr);
> vf := VectorField(<y+z*y,-x-z*x,x*y*z>,cartesian[x,y,z]);
> fieldplot3d(vf,x=-1..1,y=-1..1,z=-1..1,thickness=3);
> Curl(vf);
> fieldplot3d(Curl(vf),x=-1..1,y=-1..1,z=-1..1);
```

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);
```