Sonnenuhren

Mathematica-Programme zum Kursmaterial

!!!===> Unten sind die Vorschläge zu Modulen zu finden <===!!!

von Rolf Wirz - Version 15.2.2001 - Copyright by Rolf Wirz

■ Titel (Abfolge)

==> Vorschlag für Module
==> Putz: Alle Daten löschen, Neubeginn
==> Programm (* INITIALISATION *)
==> dataEntry: Stellt notwendige Parameter und globale Variablen zur Verfügung
==> Programm (* DEFINITION und INITIALISATION *)
==> Beispiel
==> Programm (* DEFINITION*)
==> Beispiele (* INITIALISATION *)
==> ombrixGzentr, ombrixGzentrDeltaStd: Zeichnet Ombrix-Zifferblatt...
==> Programm (* DEFINITION*)
==> Beispiele (* INITIALISATION *)
==> Kalendermodul: Berechnet Anzahl Tage des eingegebenen Datums ab...
==> Programm (* DEFINITION*)
==> Beispiele (* INITIALISATION *)
==> zeitGleichung: Berechnet Zeitgleichung und stellt Funktionen zur Verfügung:
==> Programm (* DEFINITION*)
==> Beispiele (* INITIALISATION *)
==> deklin: Deklinationsbestimmung zum Datum in Altgrad,....
==> Programm (* DEFINITION*)
==> Beispiele (* INITIALISATION *)
==> dekPlot: Plot von n Deklinationslinien,...
==> Programm (* DEFINITION*)
==> Beispiel (* INITIALISATION *)
==> zeitSchlaufen Plot von n Zeitschlaufen für Stunden...
==> Programm (* DEFINITION*)
==> Beispiele (* INITIALISATION *)
==> Drehungen, Wandtranslation in Horizontallage
==> Programm (* DEFINITION*)
==> Beispiel (* INITIALISATION *)
==> Programm (* DEFINITION*)
==> Beispiele (* INITIALISATION *)
==> Plot-Programm
==> Programm
==> Beispiele
==> Work
==> Der Modularisierung des nachfolgenden Codes ist noch nicht erarbeitet worden. Man kann sich hier fragen, welchen Output man haben möchte... Achtung! Der Run dieses Programms könnte lange dauern .....

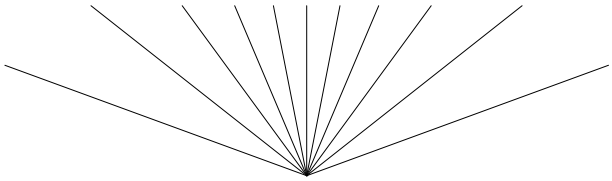
■ "Putzmaschine" einsetzen

▮ Employer la "machine de nettoyage"

```
(* Old Form: Remove["Global`@*"] *)

Remove["Global`*"]
```

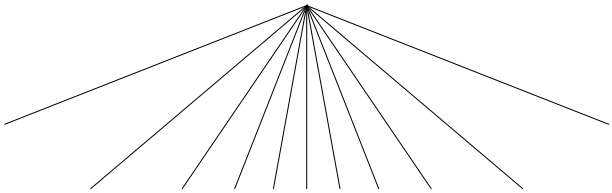
■ Horizontales Zifferblatt, Ombrix

```
(*Geographische Breite des Standorts der Uhr,
 hier Biel,BE,CH.φ erzeugt das kleine griechische phi*)
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*)  (* Remove["Global`*"];*)
(* ===> INPUT, Altgrad *)     φ = 47.09; g = 1;
(* MP berechnen,Formel:*)
hour = 2 Pi / 24 (* Tage *); MP[t_, φ_] := g Tan[t hour] / Sin[(90 - φ) Degree];
(*MP berechnen für die Werte -5 h,-4 h,...,0 h,...,5 h*)
(* ===> INPUT, t Stunden (h) *);
 tab1[φ_] := Table[MP[t, φ], {t, -5, 5, 1}]; tab1[φ];
(*Koordinaten der Punkte P berechnen für die Werte-5 h,-4 h,...,0 h,...,5 h*)
tab2[φ_] := Table[{tab1[φ][[n]], g / Tan[(90 - φ) Degree]}, {n, 1, Length[ tab1[φ]]}];
tab2[φ];
(* Ursprung versetzt, vgl. Gnomonstab parallel Erdachse! *)
ursprung[φ_] := { 0, -g / Tan[φ Degree]};(*Zifferblatt ploten:*)
p1 =
  Show[Graphics[Table[Line[{ursprung[φ], tab2[φ][[n]]}], {n, 1, Length[ tab1[φ]]}]],
   AspectRatio → Automatic];
```



■ Vertikales Zifferblatt, Ost-West-Wand, Ombrix

```
(* ===> INPUT, Altgrad *)     φ = 47.09 - 90;
hour = 2 Pi / 24 (* Tage *);
(* MP[t,φ], tab1[φ], tab2[φ] definiert im letzten Programm *)
p2 =
  Show[Graphics[Table[Line[{ursprung[φ], tab2[φ][[n]]}], {n, 1, Length[ tab1[φ]]}]],
   AspectRatio → Automatic];
```

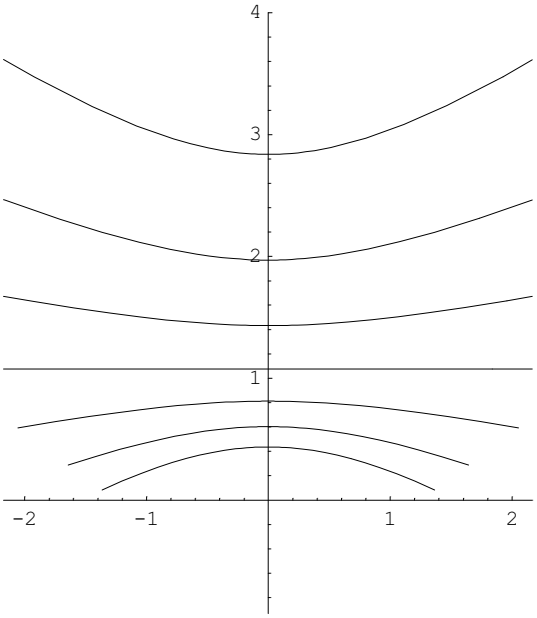


```
(* Remove[g,fi,d1,d2,d3,d4,d5,d6,v] *)
```

■ Deklinationslinien, horizontales Zifferblatt, Plot

```
(* ===> INPUT, Altgrad *)     φ = 47.09;
Print[{"Test: 180 Degree = ",  180 Degree // N}];
g = 1;    hour = 2 Pi / 24 (* Tage *);
fi = (90 - Abs[φ]);
d1 = 23.5; d2 = 16; d3 = 8; d4 = 0; d5 = -8; d6 = -16; d7 = -23.5;
(* Ursprung versetzt, vgl. Gnomonstab parallel Erdachse! *)
ursprung[φ_] := { 0, -g / Tan[φ Degree]};
v[d_, fi_, t_] :=
  g / (Cos[d Degree] Cos[t hour] Sin[fi Degree] + Sin[d Degree] Cos[fi Degree])
   {(Cos[d Degree] Sin[t hour]), Sign[φ]
     ((Cos[d Degree] Cos[t hour] Cos[fi Degree] - Sin[d Degree] Sin[fi Degree]) )};
p2 = ParametricPlot[Evaluate[{v[d1, fi, t], v[d2, fi, t],
     v[d3, fi, t], v[d4, fi, t], v[d5, fi, t],
     v[d6, fi, t], v[d7, fi, t]}], {t, -4.1, 4.1}, AspectRatio → Automatic,
   PlotRange → {ursprung[φ][[2]], 4}];
```

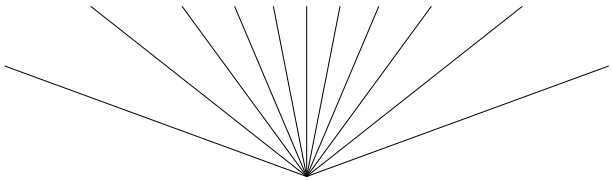
```
{Test: 180 Degree = , 3.14159}
```



■ Kombination Deklinationslinien und horizontaler Ombrix

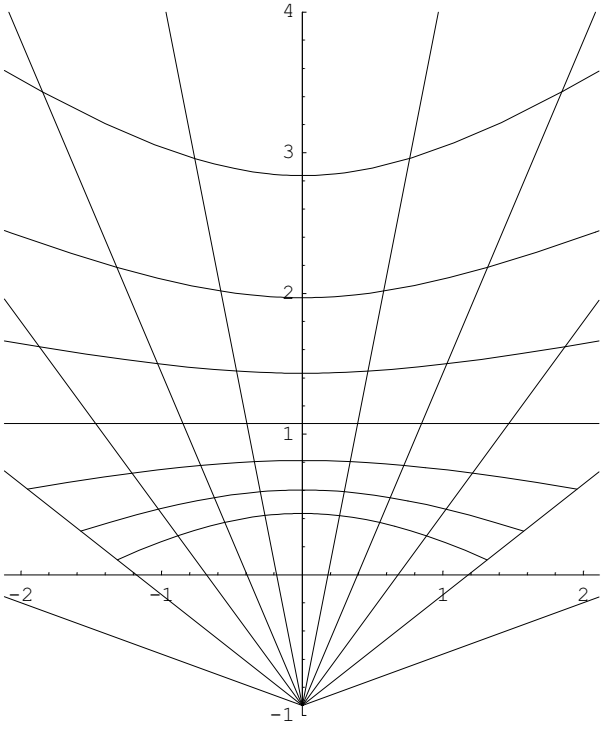
■ Ombrix (auf obige Daten wird zurückgegriffen)

```
Show[Graphics[Table[Line[{ursprung[φ], tab2[φ][[n]]}], {n, 1, Length[ tab1[φ]]}]],
  AspectRatio → Automatic];
```



■ Overlay (auf obige Daten wird zurückgegriffen)

```
p3 = ParametricPlot[
   Evaluate[{v[d1, fi, t], v[d2, fi, t], v[d3, fi, t], v[d4, fi, t], v[d5, fi, t],
     v[d6, fi, t], v[d7, fi, t]}], {t, -4.0, 4.0},
   AspectRatio → Automatic, PlotRange → {-1, 4},
   Epilog -> {Table[Line[{ursprung[φ], 4 (tab2[φ][[n]] - ursprung[φ]) + ursprung[φ]}],
      {n, 1, Length[ tab1[φ]]}]}];
```



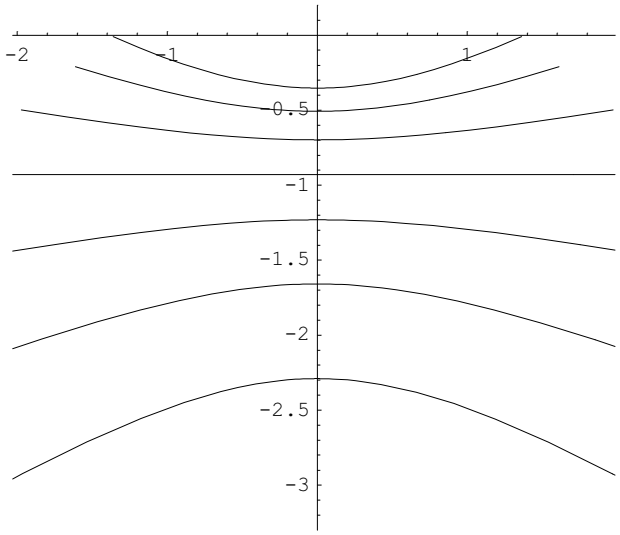

```
ursprung[φ]
```

```
{0, -0.929583}
```

■ Kombination Deklinationslinien und horizontaler Ombrix, senkrechte Wand

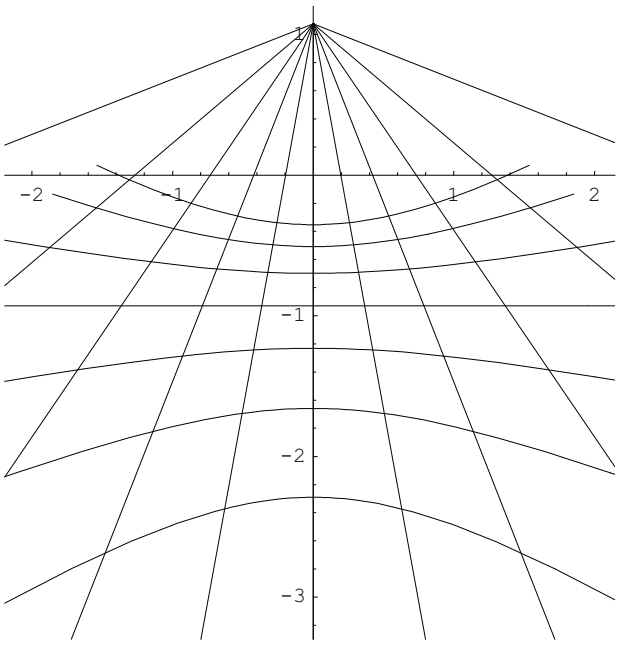
■ Deklinationslinien

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*) Remove["Global`*"];
(* ===> INPUT, Altgrad *)   φ = 47.09 - 90;
d1 = 23.5 ; d2 = 16 ; d3 = 8 ; d4 = 0 ; d5 = -8 ; d6 = -16 ; d7 = -23.5 ;
fi = (90 - Abs[φ]);     g = 1;
(* Definitionen und Rechnungen *)
hour = 2 Pi / 24 (* Tage *);
v[d_, fi_, t_] := g / (Cos[Sign[φ] d Degree] Cos[t hour] Sin[fi Degree] +
      Sin[Sign[φ] d Degree] Cos[fi Degree]) {(Cos[Sign[φ] d Degree] Sin[t hour]),
    Sign[φ] ((Cos[Sign[φ] d Degree] Cos[t hour] Cos[fi Degree] -
        Sin[Sign[φ] d Degree] Sin[fi Degree]) )};
ParametricPlot[Evaluate[{v[d1, fi, t], v[d2, fi, t], v[d3, fi, t],
    v[d4, fi, t], v[d5, fi, t],
    v[d6, fi, t], v[d7, fi, t]}], {t, -4.1, 4.1}, AspectRatio → Automatic,
  PlotRange → {-3.3, 0.2}];
```



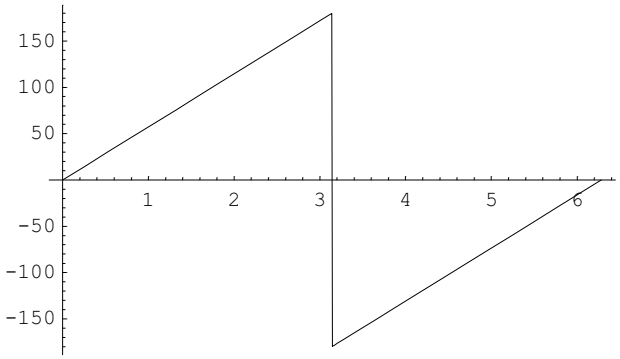
■ Overlay

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*) Remove["Global`*"];
(* ===> INPUT, Altgrad *)   φ = 47.09 - 90;
d1 = 23.5 ; d2 = 16 ; d3 = 8 ; d4 = 0 ; d5 = -8 ; d6 = -16 ; d7 = -23.5 ;
fi = (90 - Abs[φ]);     g = 1;
(* Definitionen und Rechnungen *)
hour = 2 Pi / 24 (* Tage *); MP[t_, φ_] := g Tan[t hour] / Sin[(90 - φ) Degree];
 tab1[φ_] := Table[MP[t, φ], {t, -5, 5, 1}]; tab1[φ];
tab2[φ_] := Table[{tab1[φ][[n]], g / Tan[(90 - φ) Degree]}, {n, 1, Length[ tab1[φ]]}];
tab2[φ];
v[d_, fi_, t_] := g / (Cos[Sign[φ] d Degree] Cos[t hour] Sin[fi Degree] +
      Sin[Sign[φ] d Degree] Cos[fi Degree]) {(Cos[Sign[φ] d Degree] Sin[t hour]),
    Sign[φ] ((Cos[Sign[φ] d Degree] Cos[t hour] Cos[fi Degree] -
        Sin[Sign[φ] d Degree] Sin[fi Degree]) )};
(* Ursprung versetzt, vgl. Gnomonstab parallel Erdachse! *)
ursprung[φ_] := { 0, -g / Tan[φ Degree]};
ParametricPlot[
  Evaluate[{v[d1, fi, t], v[d2, fi, t], v[d3, fi, t], v[d4, fi, t], v[d5, fi, t],
    v[d6, fi, t], v[d7, fi, t]}], {t, -4.4, 4.4},
  AspectRatio → Automatic, PlotRange → {-3.3, 1.2},
  Epilog -> {Table[Line[ {ursprung[φ],
       3.7 (tab2[φ][[n]] - ursprung[φ]) + ursprung[φ]}], {n, 1, Length[ tab1[φ]]}]}];
```



■ Ausdehnung des Arcus

```
a[x_, y_] := 180 / Pi ArcTan[y / x] + 90 Sign[y] (1 - Sign[x]);
x[t_] := Cos[t]; y[t_] := Sin[t];
p4 = Plot[Evaluate[a[x[t], y[t]]], {t, 0, 2 Pi}];
```



■ Kalenderprogramm

```
(* Monatslängen *)
m[n_] := 31; m[2] = 28; m[4] = 30; m[6] = 30; m[9] = 30; m[11] = 30; m[13] = 1;
(* Kontrolle *)
Print[Sum[m[n], {n, 1, 12}]] ;
(* Anzahl Tage seit 31.12. letzten Jahres *)
tage[tag_, mon_] := Sum[m[n], {n, 1, mon}] - m[mon] + tag;
{{"Test", "21.3.", tage[21, 3]}, {"Test", "29.8.", tage[29, 8]},
 {"Test", "21.12.", tage[21, 12]}, {"Test", "31.12.", tage[31, 12]}}
```

```
365
```

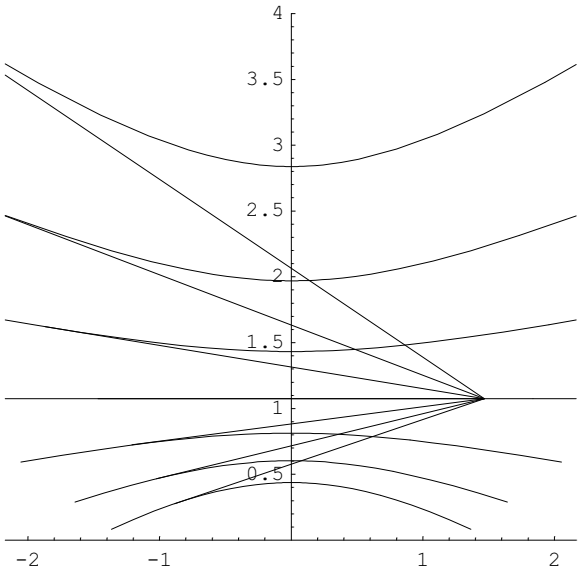
```
{{Test, 21.3., 80}, {Test, 29.8., 241}, {Test, 21.12., 355}, {Test, 31.12., 365}}
```

```
(* Monatslängen *)  Remove[m];
m[n_] := 31; m[2] = 28; m[4] = 30; m[6] = 30; m[9] = 30; m[11] = 30; m[14] = 28;
(* Für Schaltjahre *)  m[-1] = 1;
(* Anzahl Tage seit 31.12. letzten Jahres *)
tage[tag_, mon_] := Sum[m[n], {n, 1, mon}] - m[mon] + tag;
(*Anzahl Tage seit Frühlingsbeginn*)
 fpTage[tag_, mon_] := tage[tag, mon] - tage[21, 3];

(* Test des Programms: *)

{{"Test", "21.3.", fpTage[21, 3]}, {"Test", "29.8.", fpTage[29, 8]},
 {"Test", "21.12.", fpTage[21, 12]}, {"Test", "31.12.", fpTage[20, 15]},
 {"Test", "21.3.", tage[21, 3]}, {"Test", "29.8.", tage[29, 8]},
 {"Test", "21.12.", tage[21, 12]}, {"Test", "31.12.", tage[31, 12]}}
```

```
{{Test, 21.3., 0}, {Test, 29.8., 161}, {Test, 21.12., 275}, {Test, 31.12., 364},
 {Test, 21.3., 80}, {Test, 29.8., 241}, {Test, 21.12., 355}, {Test, 31.12., 365}}
```

■ Deklinationslinien und Tangenten

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*) Remove["Global`*"];
(* ===> INPUT, Altgrad *) φ = 47.09; fi = (90 - Abs[φ]); g = 1;
d[1] = 23.5 ; d[2] = 16 ; d[3] = 8 ; d[4] = 0 ; d[5] = -8 ; d[6] = -16 ; d[7] = -23.5 ;
hour = 2 Pi / 24;
v[d_, fi_, t_] := g / (Cos[Sign[φ] d Degree] Cos[t hour] Sin[fi Degree] +
     Sin[Sign[φ] d Degree] Cos[fi Degree]) {(Cos[Sign[φ] d Degree] Sin[t hour]),
   Sign[φ] ((Cos[Sign[φ] d Degree] Cos[t hour] Cos[fi Degree] -
       Sin[Sign[φ] d Degree] Sin[fi Degree]) )};
                    (* Achtung! In v[d,fi,t] wird die
  siderische Zeit benutzt! *)
ParametricPlot[Evaluate[Table[v[d[k], fi, t], {k, 1, 7}]],
  {t, -4.1, 4.1}, AspectRatio → Automatic, PlotRange → {0, 4},
  Epilog -> {
    Table[Line[{v[d[k], fi, -3], v[d[4], fi, 3]}], {k, 1, 7}]
   }];
```

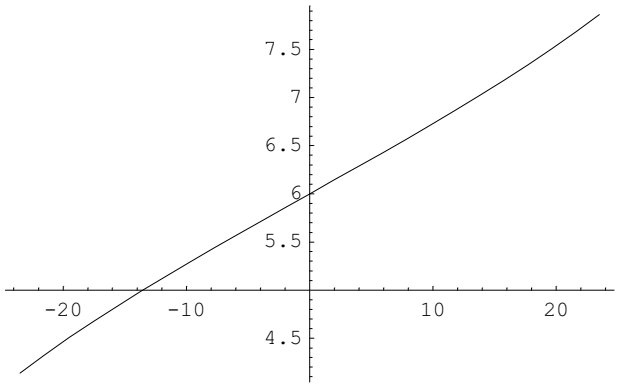


■ Berechnung der Sonnenuntergangszeit ab Mittag bei grösster und kleinster Deklination

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*) Remove["Global`*"];
hour = 2 Pi / 24;   t[δ_, ϕ_] := ArcCos[-Tan[δ Degree] Tan[ϕ Degree]] / hour;
(* Test  ===> INPUT, Altgrad *)
{{23.5, t[23.5 , 47.09 ]} , {0, t[0 , 47.09 ]} , {-23.5, t[-23.5 , 47.09 ]} }
```

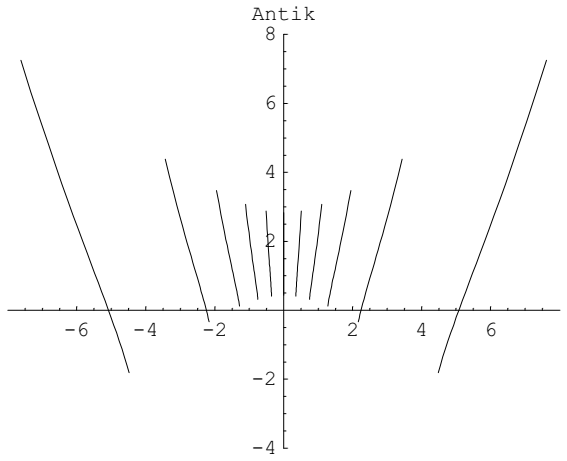
```
{{23.5, 7.85922}, {0, 6}, {-23.5, 4.14078}}
```

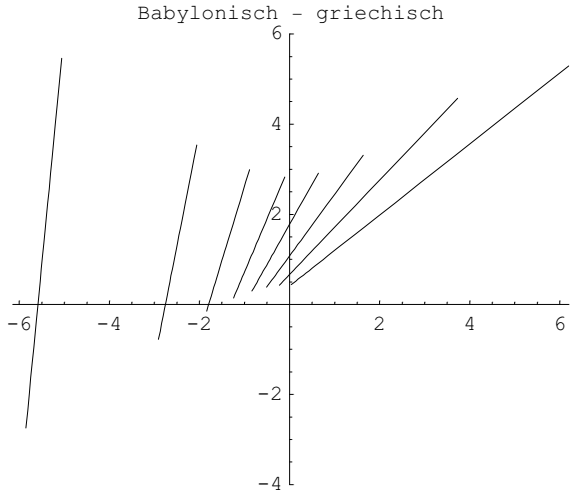
```
Plot[t[x ,  47.09 ] , {x, -23.5, 23.5}];
```

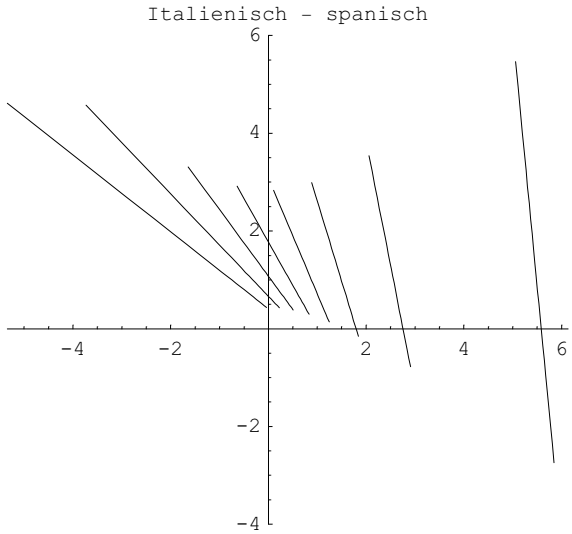


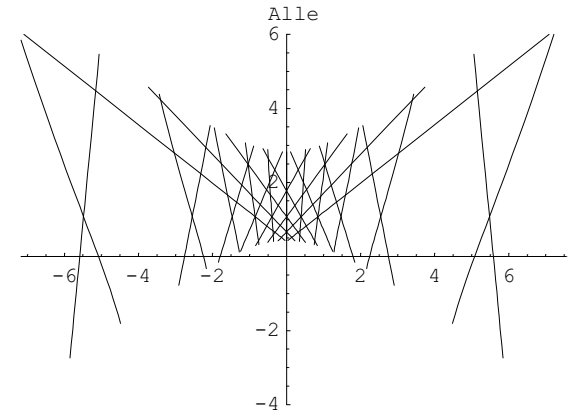
■ Linien für antike, babylinisch-griechische und italienisch-spanische Stunden

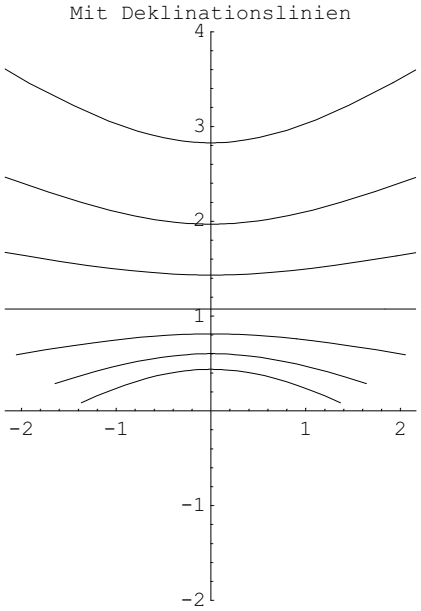
```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*) Remove["Global`*"];
(* ===> INPUT, Altgrad *)
φ = 47.09; g = 1;
fi = (90 - Abs[φ]); (* Print["fi  ",Fi]; *)
hour = 2 Pi / 24 (* Tage *);(* Print["hour  ",hour]; *)
tRad[δ_, ϕ_] := ArcCos[-Tan[δ Degree] Tan[ϕ Degree]];
(* Print["tRad  ",tRad[23.4393, φ]]; *)
t[δ_, ϕ_] := ArcCos[-Tan[δ Degree] Tan[ϕ Degree]] / hour;
(* Print["t  ",t[23.4393, φ]]; *)
tAntik[δ_, ϕ_, n_] := -t[δ, ϕ] + n (2 t[δ, ϕ]) / 12;
(* Print["tAntik  ",tAntik[23.4393, φ,3]]; *)
tBab[δ_, ϕ_, n_] := -t[δ, ϕ] + n ;
(* Print["tBab  ",tBab[23.4393, φ,3]]; *)
tIt[δ_, ϕ_, n_] := t[δ, ϕ] - 24 + n ;
(* Print["tIt  ",tIt[23.4393, φ,3]]; *)
v[δ_, fi_, t_] :=
  g / (Cos[δ Degree] Cos[t hour] Sin[fi Degree] + Sin[δ Degree] Cos[fi Degree])
   {(Cos[δ Degree] Sin[t hour]), Sign[φ]
     ((Cos[δ Degree] Cos[t hour] Cos[fi Degree] - Sin[δ Degree] Sin[fi Degree]) )};
(* Print["v  ",v[23.4393,fi,3]]; *)
plotAntik[δ_, ϕ_, n_] := v[δ, fi, tAntik[δ, ϕ, n]];
(* Print["plotAntik  ",plotAntik[23.4393,fi,3]]; *)
plotBab[δ_, ϕ_, n_] := v[δ, fi, tBab[δ, ϕ, n]];
(* Print["plotBab  ",plotBab[23.4393,fi,3]]; *)
plotIt[δ_, ϕ_, n_] := v[δ, fi, tIt[δ, ϕ, n]];
(* Print["plotIt  ",plotIt[23.4393,fi,18]]; *)
ursprung[φ_] := { 0, -g / Tan[φ Degree]};
menge1 = {plotAntik[δ, φ, 1], plotAntik[δ, φ, 2], plotAntik[δ, φ, 3],
   plotAntik[δ, φ, 4], plotAntik[δ, φ, 5], plotAntik[δ, φ, 6], plotAntik[δ, φ, 7],
   plotAntik[δ, φ, 8], plotAntik[δ, φ, 9], plotAntik[δ, φ, 10], plotAntik[δ, φ, 11]};
menge2 = {plotBab[δ, φ, 1], plotBab[δ, φ, 2], plotBab[δ, φ, 3],
   plotBab[δ, φ, 4], plotBab[δ, φ, 5], plotBab[δ, φ, 6], plotBab[δ, φ, 7],
   plotBab[δ, φ, 8], plotBab[δ, φ, 8], plotBab[δ, φ, 8], plotBab[δ, φ, 8]};
menge3 = {plotIt[δ, φ, 16], plotIt[δ, φ, 17], plotIt[δ, φ, 18], plotIt[δ, φ, 19],
   plotIt[δ, φ, 20], plotIt[δ, φ, 21], plotIt[δ, φ, 22], plotIt[δ, φ, 23]};
menge4 = Join[menge1, menge2, menge3];
menge5 = Join[menge4, {v[d1, fi, t], v[d2, fi, t], v[d3, fi, t], v[d4, fi, t], v[d5, fi, t
    v[d6, fi, t], v[d7, fi, t]}];
p1 = ParametricPlot[Evaluate[menge1], {δ, -23.4393, 23.4393},
   AspectRatio → Automatic, PlotRange → {-4, 8}, PlotLabel → "Antik"];
p2 = ParametricPlot[Evaluate[menge2], {δ, -23.4393, 23.4393}, AspectRatio → Automatic,
   PlotRange → {-4, 6}, PlotLabel → "Babylonisch - griechisch"];
p3 = ParametricPlot[Evaluate[menge3], {δ, -23.4393, 23.4393}, AspectRatio → Automatic,
   PlotRange → {-4, 6}, PlotLabel → "Italienisch - spanisch"];
p4 = ParametricPlot[Evaluate[menge4], {δ, -23.4393, 23.4393},
   AspectRatio → Automatic, PlotRange → {-4, 6}, PlotLabel → "Alle"];
d1 = 23.4393 ; d2 = 16 ; d3 = 8 ; d4 = 0 ; d5 = -8 ; d6 = -16 ; d7 = -23.4393 ;
p5 = ParametricPlot[
   Evaluate[{v[d1, fi, t], v[d2, fi, t], v[d3, fi, t], v[d4, fi, t], v[d5, fi, t],
     v[d6, fi, t], v[d7, fi, t]}], {t, -4.1, 4.1},
   AspectRatio → Automatic, PlotRange → {-2, 4},
   PlotLabel → "Mit Deklinationslinien"];
Show[p4, p5];
```

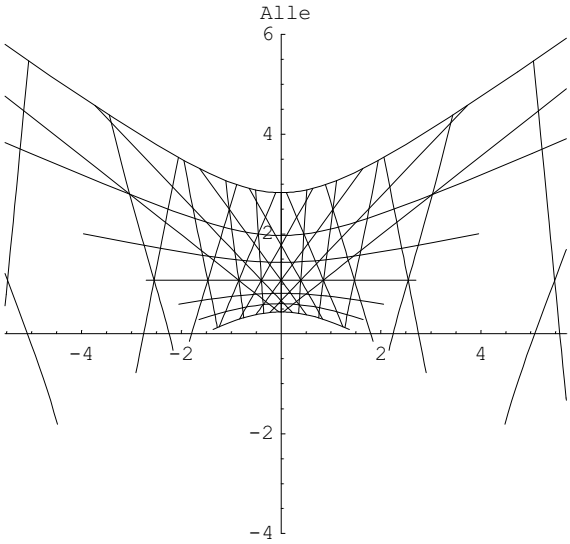

Antik
8
6
4
2
-2
-4
-6
-4
-2
2
4
6


Babylonisch - griechisch
6
4
2
-2
-4
-6
-4
-2
2
4
6


Italienisch - spanisch
6
4
2
-2
-4
-4
-2
2
4
6

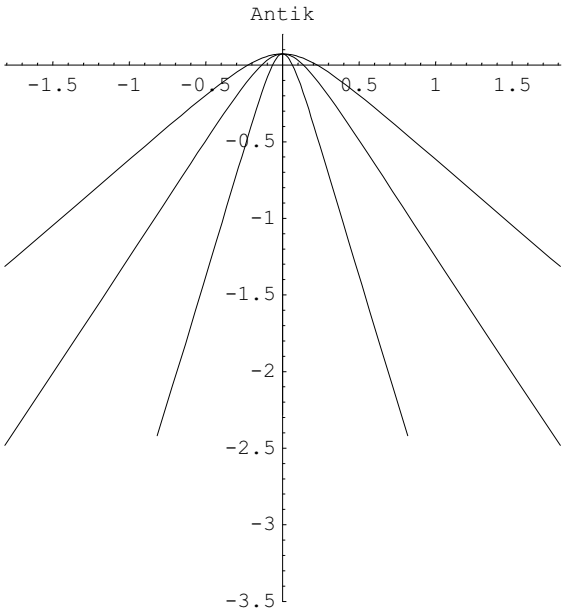

Alle
6
4
2
-6
-4
-2
2
4
6
-2
-4

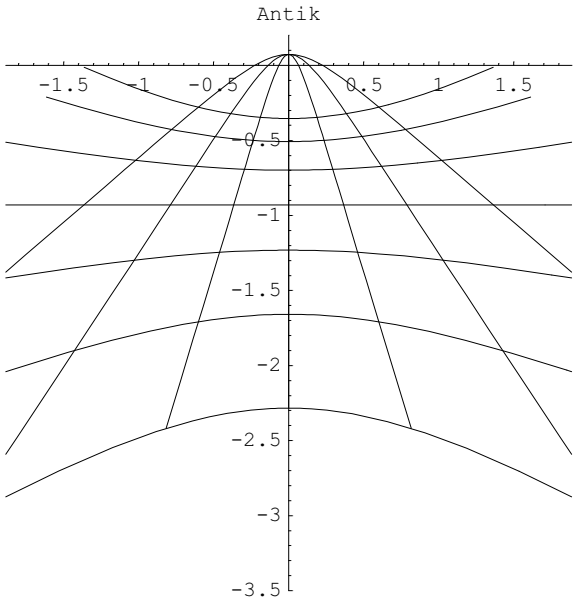

Mit Deklinationslinien
4
3
2
1
-2
-1
1
2
-1
-2

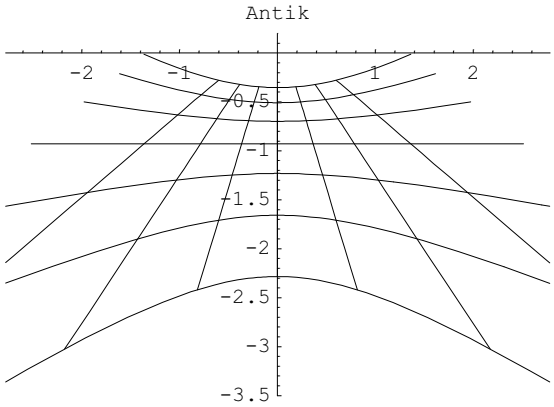

Alle
6
4
2
-4
-2
2
4
-2
-4

■ Antike Stundenlinien (keine Geraden)

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*) Remove["Global`*"];
(* ===> INPUT, Altgrad *)
(* φ=47.09; *) (*horizontal*)
φ = 47.09 - 90;  (*vertikal*)
g = 1;
fi = (90 - Abs[φ]);(* Print["fi  ",Fi]; *)
hour = 2 Pi / 24 (* Tage *);(* Print["hour  ",hour]; *)
tRad[δ_, ϕ_] := ArcCos[-Tan[δ Degree] Tan[ϕ Degree]];
(* Print["tRad  ",tRad[23.4393, φ]]; *)
t[δ_, ϕ_] := ArcCos[-Tan[δ Degree] Tan[ϕ Degree]] / hour;
(* Print["t  ",t[23.4393, φ]]; *)
tAntik[δ_, ϕ_, n_] := -t[δ, ϕ] + n (2 t[δ, ϕ]) / 12;
(* Print["tAntik  ",tAntik[23.4393, φ,3]]; *)
v[δ_, fi_, t_] :=
  g / (Cos[δ Degree] Cos[t hour] Sin[fi Degree] + Sin[δ Degree] Cos[fi Degree])
   {(Cos[δ Degree] Sin[t hour]), Sign[φ]
     ((Cos[δ Degree] Cos[t hour] Cos[fi Degree] - Sin[δ Degree] Sin[fi Degree]) )};
(* Print["v  ",v[23.4393,fi,3]]; *)
plotAntik[δ_, ϕ_, n_] := v[δ, fi, tAntik[δ, ϕ, n]];
ursprung[φ_] := { 0, -g / Tan[φ Degree]};
menge1 = {(*plotAntik[δ,φ,1],plotAntik[δ,φ,2],*)
   plotAntik[δ, φ, 3], plotAntik[δ, φ, 4], plotAntik[δ, φ, 5],
   plotAntik[δ, φ, 6], plotAntik[δ, φ, 7], plotAntik[δ, φ, 8], plotAntik[δ, φ, 9]
   (*,plotAntik[δ,φ,10],plotAntik[δ,φ,11]*)};
ϑ = 47.09;(*Plotbereich*)
p0 = ParametricPlot[Evaluate[menge1], {δ, -23.4393, ϑ},
   AspectRatio → Automatic, PlotRange → {-3.5, 0.2}, PlotLabel → "Antik"];
p1 = ParametricPlot[Evaluate[menge1], {δ, -23.4393, 23.4393}, AspectRatio → Automatic,
   PlotRange → {-3.5, 0.2}, PlotLabel → "Antik", DisplayFunction → Identity];
d1 = 23.4393 ; d2 = 16 ; d3 = 8 ; d4 = 0 ; d5 = -8 ; d6 = -16 ; d7 = -23.4393 ;
p2 = ParametricPlot[
   Evaluate[{v[d1, fi, t], v[d2, fi, t], v[d3, fi, t], v[d4, fi, t], v[d5, fi, t],
     v[d6, fi, t], v[d7, fi, t]}], {t, -4.1, 4.1},
   AspectRatio → Automatic, PlotRange → {-3.5, 0.2},
   PlotLabel → "Mit Deklinationslinien", DisplayFunction → Identity];
Show[p0, p2, DisplayFunction → $DisplayFunction];
Show[p1, p2, DisplayFunction → $DisplayFunction];
```

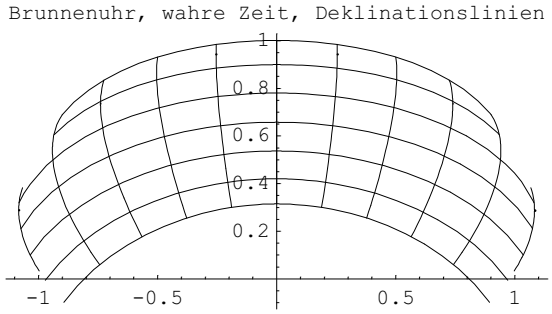

Antik
-1.5
-1
-0.5
0.5
1
1.5
-0.5
-1
-1.5
-2
-2.5
-3
-3.5


Antik
-1.5
-1
-0.5
0.5
1
1.5
-0.5
-1
-1.5
-2
-2.5
-3
-3.5


Antik
-2
-1
1
2
-0.5
-1
-1.5
-2
-2.5
-3
-3.5

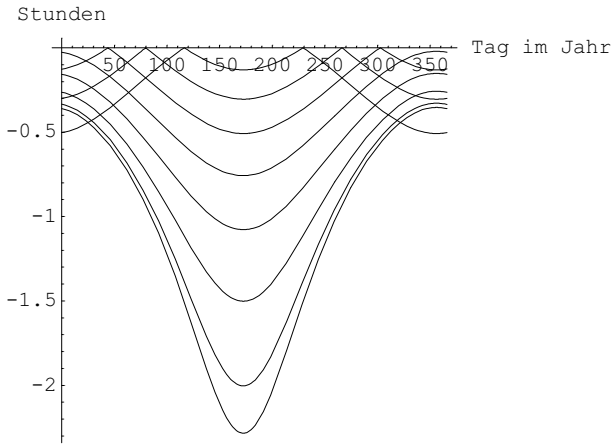
■ Brunnensonnenuhr

```
(*"Putzmaschine"*)(*Old Form:Remove["Global`@*"]*)Remove["Global`*"];
φ = 47.09; g = 1;
hour = 2 Pi / 24 (*Tage*);
MP[t_, φ_] := g Tan[t hour] / Sin[(90 - φ) Degree];
tab1[φ_] := Table[MP[t, φ], {t, -5, 5, 1}]; tab1[φ];
tab2[φ_] := Table[{tab1[φ][[n]], g / Tan[(90 - φ) Degree]}, {n, 1, Length[tab1[φ]]}];
tab2[φ];
ursprung[φ_] := {0, -g / Tan[φ Degree]};
fi = (90 - Abs[φ]);
d1 = 23.5; d2 = 16; d3 = 8; d4 = 0; d5 = -8; d6 = -16; d7 = -23.5;
v[d_, fi_, t_] :=
  g / (Cos[d Degree] Cos[t hour] Sin[fi Degree] + Sin[d Degree] Cos[fi Degree])
   {(Cos[d Degree] Sin[t hour]),
    Sign[φ] ((Cos[d Degree] Cos[t hour] Cos[fi Degree] - Sin[d Degree] Sin[fi Degree]))};
v3[vec_] := Append[vec, -g];
len[vec1_] := Sqrt[vec1.vec1];
hvec = {0, 0, -g};
einfall[veC_] := ArcCos[veC.hvec / (len[veC] * len[hvec])];
ausfall[veq_] := ArcSin[Sin[einfall[veq]] / 1.333];
new[vq_] := vq * Tan[ausfall[v3[vq]]] / Tan[einfall[v3[vq]]];
vB[d_, fi_, t_] := new[v[d, fi, t]];
pd[t_, m_] := ParametricPlot[Evaluate[vB[d, fi, t]], {d, -m, m},
   DisplayFunction → Identity];
pt[d_, n_] :=
  ParametricPlot[Evaluate[vB[d, fi, t]], {t, -n, n}, DisplayFunction → Identity];
p1 = Show[pd[-5, 13], pd[-4, 23.5], pd[-3, 23.5], pd[-2, 23.5],
   pd[-1, 23.5], pd[0, 23.5], pd[1, 23.5], pd[2, 23.5],
   pd[3, 23.5], pd[4, 23.5], pd[5, 13], AspectRatio → Automatic,
   PlotLabel → "Brunnenuhr, wahre Zeit, Deklinationslinien"];
p2 = Show[pt[d1, 5], pt[d2, 5], pt[d3, 5], pt[d4, 5],
   pt[d5, 5], pt[d6, 4], pt[d7, 4], AspectRatio → Automatic,
   PlotLabel → "Brunnenuhr, wahre Zeit, Deklinationslinien"];
Show[p1, p2, DisplayFunction → $DisplayFunction];
```



■ Zylindersonnenuhr, Zifferblatt

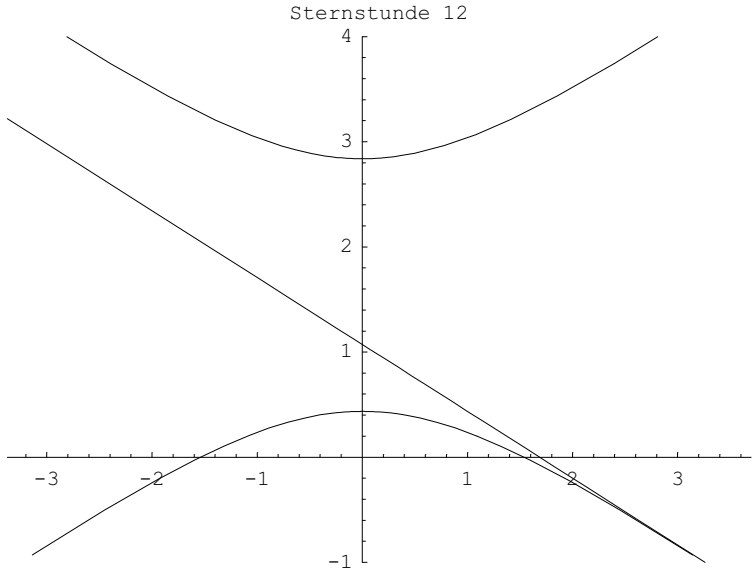
```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*)  Remove["Global`*"]
(* Konstanten *)
φ = 47.09; ϕ = (90 - φ) Degree;
g = 1; (*g=Stablänge*)
(*ϕ=47.09 Degree;g=1;*)
achsverh = 1;(*Achsenverhältnis beim Plot*)
len[v_] := Sqrt[v.v];
s1[t_, δ_, ϕ_] := {Cos[δ] Cos[t] Cos[ϕ] - Sin[δ] Sin[ϕ],
   Cos[δ] Sin[t], Cos[δ] Cos[t] Sin[ϕ] + Sin[δ] Cos[ϕ]};
s[t_, δ_, ϕ_] := s1[t, δ, ϕ] / len[s1[t, δ, ϕ]];
h1[t_, δ_, ϕ_] := {Cos[δ] Cos[t] Cos[ϕ] - Sin[δ] Sin[ϕ], Cos[δ] Sin[t], 0};
h[t_, δ_, ϕ_] := h1[t, δ, ϕ] / len[h1[t, δ, ϕ]];
α[t_, δ_, ϕ_] := ArcCos[s[t, δ, ϕ].h[t, δ, ϕ]];
d[t_, δ_, ϕ_] := -g * Tan[α[t, δ, ϕ]];
tabelle[δ_] := Evaluate[Table[d[n 2 Pi / 24, δ, ϕ], {n, 0, 4}]];
(*Print[tabelle[δ]];*)
m0 = (356.5399 - 360) Degree; e = 0.0167092;
L0 = (282.9399 - 360) Degree; Eps = 23.4393 Degree;
L01 = 0.0000470684 Degree; m01 = 0.9856002664 Degree;
(* Keplerfunktion *)
f[eE_, d_] := eE - e Sin[eE] - m0 - m01 d;
eE[d_] := eNS /. FindRoot[f[eNS, d] == 0, {eNS, 3}];
delta[d_] :=
  ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 + L01 d] Sin[Eps]];
p[n_] := Plot[Evaluate[d[n 2 Pi / 24, δ, ϕ]] /. δ -> delta[d1], {d1, 0, 366},
   AxesLabel → {"Tag im Jahr", "Stunden"}, DisplayFunction → Identity];
t = Table[p[n], {n, 0, 7}]; graph = Show[t, DisplayFunction → $DisplayFunction,
  AspectRatio → achsverh];
```

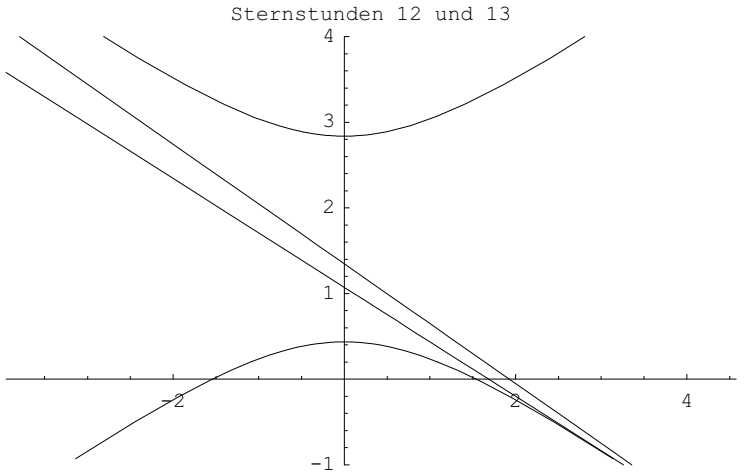


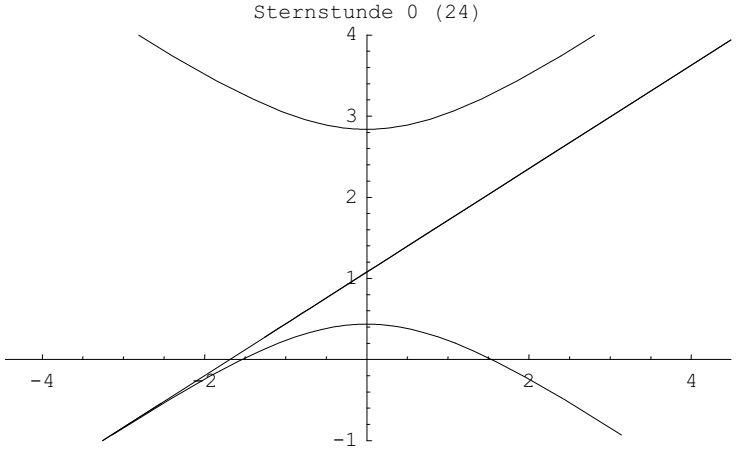

```
(* Braucht viel Rechenzeit! *)
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*)  Remove["Global`*"]
(* Monatslängen *)
```

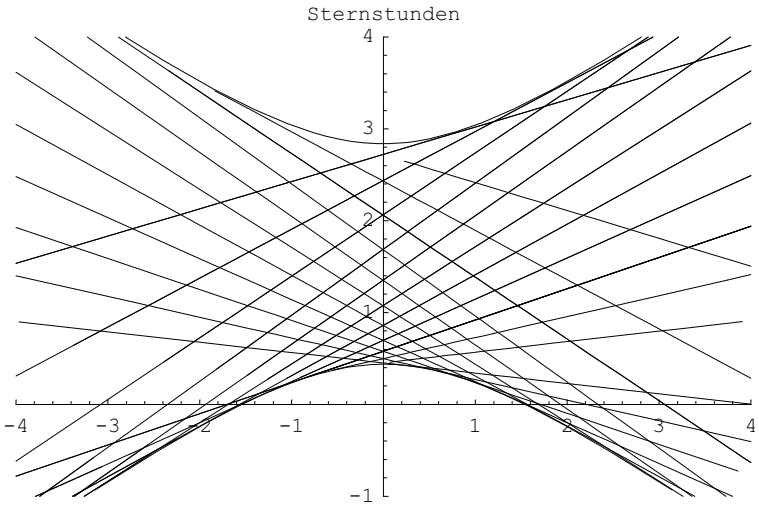
```
m[n_] := 31; m[2] = 28; m[4] = 30; m[6] = 30; m[9] = 30; m[11] = 30; m[13] = 1;
(* Anzahl Tage seit 31.12. letzten Jahres *)
tage[tag_, mon_] := Sum[m[n], {n, 1, mon}] - m[mon] + tag;
day0 = tage[21, 3];
(* Konstanten *)
m0 = (356.5399 - 360) Degree; e = 0.0167092;
L0 = (282.9399 - 360) Degree; Eps = 23.4393 Degree;
L01 = 0.0000470684 Degree; m01 = 0.9856002664 Degree;
(* Keplerfunktion *)
f[eE_, day_] := eE - e Sin[eE] - m0 - m01 day;
eE[day_] := eNS /. FindRoot[f[eNS, day] == 0, {eNS, 3}];
zG[day_] :=
  ArcTan[Tan[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[day] / 2 ]] + L0 + L01 day] Cos[Eps]] /
    (2 Pi) 24;
zG1[day_] := zG[day] + 6;
zG2[day_ ] := Ceiling[(zG[day] - zG[day + 1]) / 12];
zG3[day_ ] := Sum[zG2[n], {n, 0, Floor[day]}];
zG4[day_ ] := zG1[day] + 12 zG3[day] - zG1[day0];
(* Plot[zG4[day],{day,0,480},PlotLabel→"AR"]; *)
(* Rektaszension im Stundenmass *)
AR[day_ ] := zG4[day];
(* Sternzeit im Stundenmass, t in Stunden *)tStern[day_, t_] := t + AR[day];
(* Keplerfunktion *)
f[eE_, day_] := eE - e Sin[eE] - m0 - m01 day;
eE[day_] := eNS /. FindRoot[f[eNS, day] == 0, {eNS, 3}];
(*Deklination im Grad*)delta[day_] :=
  ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[day] / 2 ]] + L0 + L01 day] Sin[Eps]] /
    (2 Pi) 360;
(* Plot[delta[day],{day,0,365}]; *)
φ = 47.09; g = 1;
hour = 2 Pi / 24 (*Tage*);
fi = (90 - Abs[φ]);
(* Plot-Vektor, δ Degree, t hour, fi Degree *)
v[δ_, fi_, t_] :=
  g / (Cos[δ Degree] Cos[t hour] Sin[fi Degree] + Sin[δ Degree] Cos[fi Degree])
   {(Cos[δ Degree] Sin[t hour]),
    Sign[φ] ((Cos[δ Degree] Cos[t hour] Cos[fi Degree] - Sin[δ Degree] Sin[fi Degree]))};
pt[δ_, n_] := ParametricPlot[Evaluate[v[δ, fi, t]], {t, -n, n},
   AspectRatio → Automatic, DisplayFunction → Identity];
plotδStern[t_, n1_, n2_] := ParametricPlot[v[delta[day], fi, tStern[day, t]],
   {day, n1, n2}, AspectRatio → Automatic,
   PlotLabel → "Sternstunden",
   DisplayFunction → Identity];
Show[plotδStern[0, 25, 180],
  pt[23.5, 6], pt[-23.5, 4], AspectRatio → Automatic, PlotRange → {-1, 4},
  DisplayFunction → $DisplayFunction, PlotLabel → "Sternstunde 12"];
Show[plotδStern[0, 25, 180], plotδStern[1, 15, 170],
  pt[23.5, 6], pt[-23.5, 4], AspectRatio → Automatic, PlotRange → {-1, 4},
  DisplayFunction → $DisplayFunction, PlotLabel → "Sternstunden 12 und 13"];
Show[plotδStern[12, 25 + 182, 180 + 182],
 pt[23.5, 6], pt[-23.5, 4], PlotLabel → "Sternstunde 0 (24)", AspectRatio → Automatic,
 PlotRange → {-1, 4}, DisplayFunction → $DisplayFunction]; Show[
 plotδStern[-5, 84, 240],
 plotδStern[-4, 72, 228],
 plotδStern[-3, 60, 215],
```

```
plotδStern[-2, 49, 206],
plotδStern[-1, 37, 195],
plotδStern[0, 25, 180],
plotδStern[1, 15, 170],
plotδStern[2, 5, 150],
plotδStern[3, 0, 130],
plotδStern[3, -8, 130],
plotδStern[4, -9, 100],
plotδStern[5, 9, 80],
plotδStern[-5 + 12, 84 + 182, 240 + 182],
plotδStern[-4 + 12, 72 + 182, 228 + 182],
plotδStern[-3 + 12, 60 + 182, 215 + 182],
plotδStern[-2 + 12, 49 + 182, 206 + 182],
plotδStern[-1 + 12, 37 + 182, 195 + 182],
plotδStern[0 + 12, 25 + 182, 180 + 182],
plotδStern[1 + 12, 15 + 182, 170 + 182],
plotδStern[2 + 12, 5 + 182, 150 + 182],
plotδStern[3 + 12, 0 + 182, 130 + 182],
plotδStern[3 + 12, -8 + 182, 130 + 182],
plotδStern[4 + 12, -9 + 182, 100 + 182],
plotδStern[5 + 12, 9 + 182, 80 + 182],
pt[23.5, 6], pt[-23.5, 4], AspectRatio → Automatic,
PlotRange → {{-4, 4}, {-1, 4}}, DisplayFunction → $DisplayFunction];
```




Sternstunden 12 und 13
4
3
2
1
-1
-2
2
4


Sternstunde 0 (24)
4
3
2
1
-1
-4
-2
2
4


Sternstunden
4
3
2
1
-1
-4
-3
-2
-1
1
2
3
4

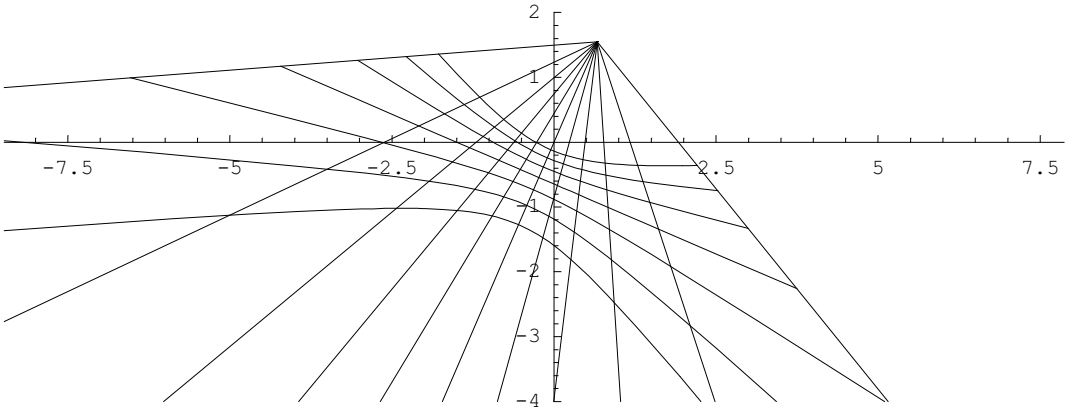
■ Schiefe Wand in schiefer Richtung: Koordinatentransformation an einen neuen Ort

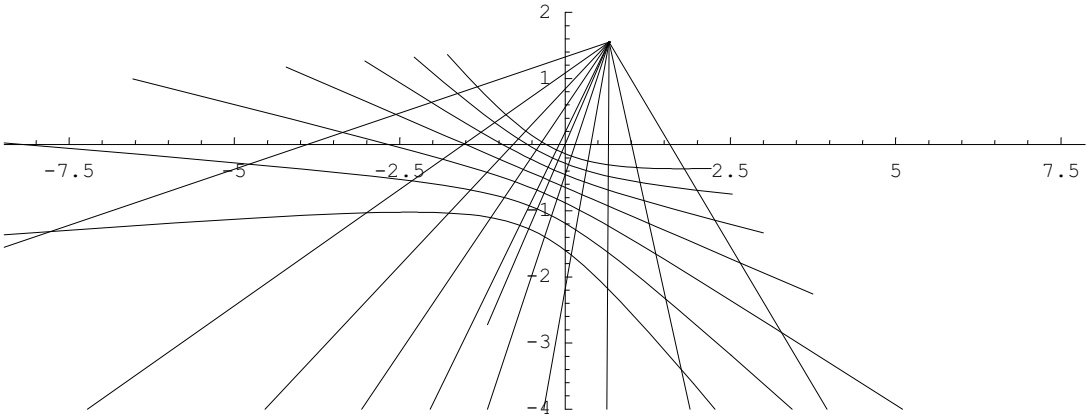
■ Umrechnung mit Vektorprodukt und Umrechnen des Vertikalvektors und zur Drehung des Outputs in Vertikallage in Einzelschritten

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*) (*  Remove["Global`*"]; *)
(* Umrechnung mit Vektorprodukt *)
(* ===> INPUT, Altgrad *)  a = 30 ; b = 6 ;  φ = 47.09; φ0 = φ;
dek[1] = 23.5 ; dek[2] = 16 ; dek[3] = 8 ; dek[4] = 0 ;
dek[5] = -8 ; dek[6] = -16 ; dek[7] = -23.5 ;  g = 1;
u = {Cos[a Degree], Sin[a Degree], 0};
v = {-Sin[a Degree] Sin[b Degree], Cos[a Degree] Sin[b Degree], Cos[b Degree]};
w = Transpose[{Cross[u, v]}];
d = (90 - φ) ;
dreh =
  {{1, 0, 0}, {0, Cos[d Degree], -Sin[d Degree]}, {0, Sin[d Degree], Cos[d Degree]}};
r = dreh.w; r1 = Flatten[r];
R = {- r1[[2]], -r1[[1]], r1[[3]]};
Delta1 = ArcCos[Sqrt[(R[[1]]^2 + R[[2]]^2) / (R[[1]]^2 + R[[2]]^2 + R[[3]]^2)]] ;
Delta2 = ArcCos[Sqrt[R[[1]]^2 / (R[[1]]^2 + R[[2]]^2)]]; dY = Delta1; dZ = Delta2;
Delta1Grad = Delta1 / Degree // N; Delta2Grad = Delta2 / (2 Pi) 360 // N;;
Delta2Std = Delta2 / (2 Pi) 24 // N;
Print["Delta1Grad = ", Delta1Grad,
  ", Delta2Grad = ", Delta2Grad, ", Delta2Std = ", Delta2Std];
(*Umrechnen des Vertikalvektors und zur Drehung des
  Outputs in Vertikallage in Einzelschritten*)
Print["Test Matrixprodukt ===> ", dreh.v == Flatten[dreh.Transpose[{v}]]];
goAeq[vec_] := {-(dreh.vec)[[2]], -(dreh.vec)[[1]], (dreh.vec)[[3]]};
xKS = {1, 0, 0}; yKS = {0, 1, 0}; zKS = {0, 0, 1};
vAeq = goAeq[v]; uAeq = goAeq[u]; xAeq = goAeq[xKS];
yAeq = goAeq[yKS]; zAeq = goAeq[zKS];
drehZ = {{Cos[-dZ], -Sin[-dZ], 0}, {Sin[-dZ], Cos[-dZ], 0}, {0, 0, 1}};
drehY = {{Cos[-dY], 0, -Sin[-dY]}, {0, 1, 0}, {Sin[-dY], 0, Cos[-dY]}};
goNew[vec_] := drehZ.(drehY.vec);
xNew = goNew[xKS]; yNew = goNew[yKS]; zNew = goNew[zKS];
Print["Kontrolle 1. Koord 0 ===> ",
 solvU = Solve[uAeq == u1 xNew + u2 yNew + u3 zNew, {u1, u2, u3}] // Chop // Flatten]
Print["Kontrolle 1. Koord 0 ===> ",
  solvV = Solve[vAeq == v1 xNew + v2 yNew + v3 zNew, {v1, v2, v3}] // Chop // Flatten];
uRichtung = {u1, u2, u3} /. solvU;
vRichtung = {v1, v2, v3} /. solvV;
vLen[vec_] := Sqrt[vec.vec];
drehNewArc = ArcCos[vRichtung.zKS / (vLen[vRichtung] vLen[zKS])];
drehNewGrad = drehNewArc / (2 Pi) 360; Print["Drehung der Vertikalen ", drehNewGrad];
vv = -drehNewArc;
drehPlot = {{Cos[vv], -Sin[vv]}, {Sin[vv], Cos[vv]}};
Print["drehPlot = ", drehPlot // MatrixForm];
(* Plot *)
φ = - Delta1Grad; fi = (90 - Abs[φ]);
```

```
hour = 2 Pi / 24 (* Tage *); MP[t_, φ_] := g Tan[t hour] / Sin[(90 - φ) Degree];
tab1[φ_] := Table[MP[t, φ], {t, -5, 5, 1}]; tab1[φ];
tab2[φ_] := Table[{tab1[φ][[n]], g / Tan[(90 - φ) Degree]}, {n, 1, Length[ tab1[φ]]}];
tab2[φ];
 vV[h_, fi_, t_] := drehPlot.
    (g / (Cos[Sign[φ] h Degree] Cos[t hour] Sin[fi Degree] + Sin[Sign[φ] h Degree]
          Cos[fi Degree]) {(Cos[Sign[φ] h Degree] Sin[t hour]),
       Sign[φ] ((Cos[Sign[φ] h Degree] Cos[t hour] Cos[fi Degree] -
           Sin[Sign[φ] h Degree] Sin[fi Degree]) )});
(* vV[h_,fi_,t_]:=drehPlot. g/(Cos[d Degree] Cos[t hour] Sin[fi Degree] +
       Sin[d Degree] Cos[fi Degree])
    {(Cos[d Degree] Sin[t hour]),Sign[φ] ((Cos[d Degree] Cos[t hour]
          Cos[fi Degree] - Sin[d Degree] Sin[fi Degree]) )};*)
(* Ursprung versetzt, vgl. Gnomonstab parallel Erdachse! *)
ursprung[φ_] := { 0, -g / Tan[φ Degree]};
ParametricPlot[Evaluate[Table[vV[dek[k], fi, t], {k, 1, 7}]],
  {t, -5, 5}, AspectRatio → Automatic, PlotRange → {-4, 2} ,
  Epilog -> {Table[Line[{drehPlot.ursprung[φ], drehPlot.
        (4 ( tab2[φ][[n]] - ursprung[φ]) + ursprung[φ])}], {n, 1, Length[ tab1[φ]]}]}];
tab3[φ_] := Table[MP[(t - Delta2Std), φ], {t, -2, 7, 1}]; tab3[φ];
tab4[φ_] := Table[{tab3[φ][[n]], g / Tan[(90 - φ) Degree]}, {n, 1, Length[ tab3[φ]]}];
tab3[φ];
ParametricPlot[Evaluate[Table[vV[dek[k], fi, t], {k, 1, 7}]],
  {t, -5, 5}, AspectRatio → Automatic, PlotRange → {-4, 2} ,
  Epilog → Join[Table[Line[{drehPlot.ursprung[φ], drehPlot.
        (4 ( tab4[φ][[n]] - ursprung[φ]) + ursprung[φ])}], {n, 1, Length[ tab3[φ]]}],
    {Line[{drehPlot.ursprung[φ], 5 drehPlot.tab2[φ][[6]]}]}]];
Print["tab1[φ]:= ", tab1[φ], " Plot von -5 bis 5 Std. versetzte Zeit "];
Print["tab3[φ]:= ", tab3[φ],
 " Plot von -3-Delta2Std bis 7-Delta2Std korrigierte Zeit."]
```

```
Delta1Grad = 30.6534, Delta2Grad = 35.3121, Delta2Std = 2.35414

Test Matrixprodukt ===> True

Kontrolle 1. Koord 0 ===> {u1 → 0, u2 → -0.918372, u3 → 0.395719}

Kontrolle 1. Koord 0 ===> {v1 → 0, v2 → 0.395719, v3 → 0.918372}

Drehung der Vertikalen 23.3108

drehPlot = ( 0.918372  0.395719
            -0.395719  0.918372)
```





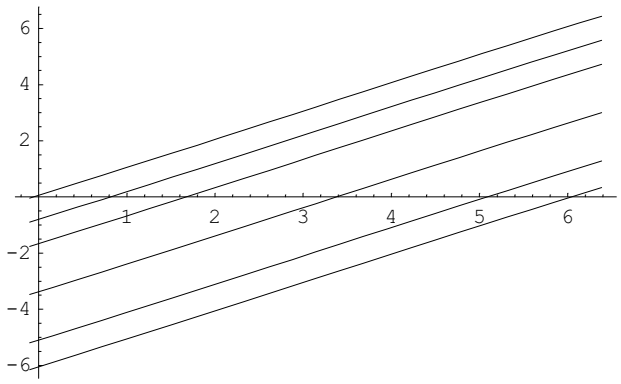
tab1[φ]:= {-4.33824, -2.01339, -1.16243, -0.671129, -0.311472, 0, 0.311472, 0.671129, 1.16243, 2.01339, 4.33824} Plot von -5 bis 5 Std. versetzte Zeit

tab3[φ]:= {-2.52871, -1.40076, -0.823416, -0.430276, -0.108084, 0.198444, 0.534359, 0.964651, 1.64102, 3.14041} Plot von -3-Delta2Std bis 7-Delta2Std korrigierte Zeit.

■ Numerische Lösung der Keplergleichung

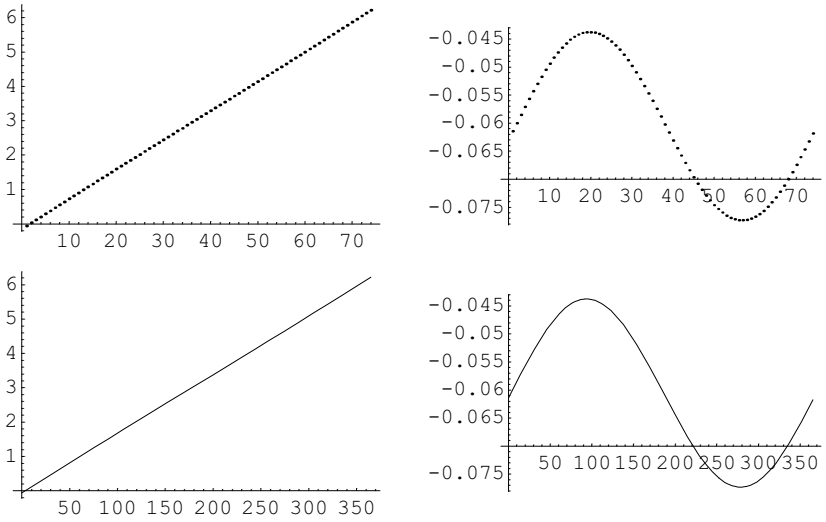
■ Keplerfunktion

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*)
Remove["Global`*"]; (* Konstanten *)
m0 = (356.5399 - 360) Degree; e = 0.0167092;
L0 = (282.9399 - 360) Degree; Eps = 23.4393 Degree;
L01 = 0.0000470684 Degree; m01 = 0.9856002664 Degree;
(* Aus Kalenderprogramm *)
m[n_] := 31; m[2] = 28; m[4] = 30; m[6] = 30;
m[9] = 30; m[11] = 30; m[11] = 30; m[14] = 28; m[-1] = 1;
(* Anzahl Tage seit 31.12. letzten Jahres *)
tage[tag_, mon_] := Sum[m[n], {n, 1, mon}] - m[mon] + tag;
day = 21; month = 12; d0 = tage[day, month] ;
(* Keplerfunktion *)
f[eE_, d_] := eE - e Sin[eE] - m0 - m01 d;
Plot[{f[x, 0], f[x, 50], f[x, 100], f[x, 200], f[x, 300], f[x, d0]},
  {x, 0 - 0.1, 2 Pi + 0.1}];
```



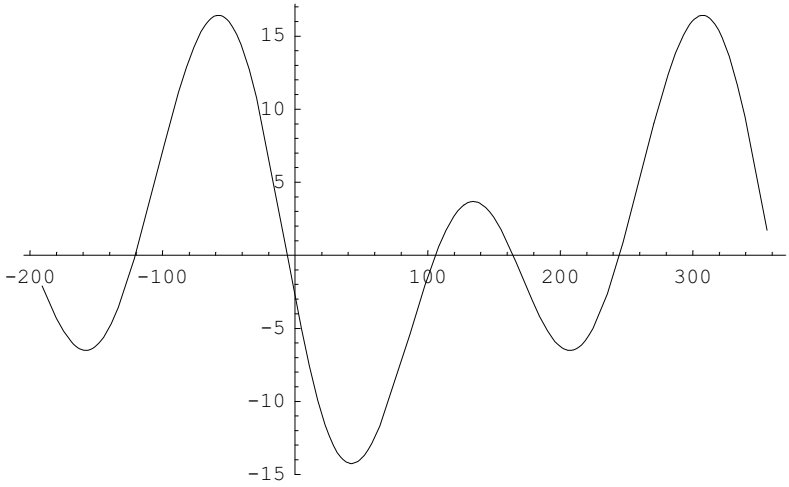
■ Keplergleichung lösen, Plot

```
(* Benützt Funktion f[eE,d] des letzten Programms *)
eE[d_] := eNS /. FindRoot[f[eNS, d] == 0, {eNS, 3}];
tab1 = Table[eE[d], {d, 0, 365.242199, 5}];
tab2 = Table[eE[d] - 2 Pi d / 365.242199, {d, 0, 365.242199, 5}];
Show[GraphicsArray[{{ListPlot[tab1, DisplayFunction → Identity],
     ListPlot[tab2, DisplayFunction → Identity]},
    {Plot[eE[d], {d, 0, 365.242199}, DisplayFunction → Identity],
     Plot[eE[d] - 2 Pi d / 365.242199, {d, 0, 365.242199}, DisplayFunction → Identity]}
   }], DisplayFunction → $DisplayFunction];
```



■ Zeitgleichung lösen

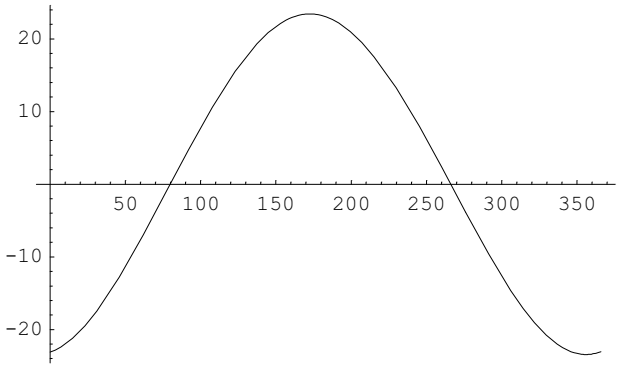
```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*)  Remove["Global`*"]
(* Konstanten *)
m0 = (356.5399 - 360) Degree; e = 0.0167092;
L0 = (282.9399 - 360) Degree; Eps = 23.4393 Degree;
L01 = 0.0000470684 Degree; m01 = 0.9856002664 Degree;
(* Keplerfunktion *)
f[eE_, d_] := eE - e Sin[eE] - m0 - m01 d;
eE[d_] := eNS /. FindRoot[f[eNS, d] == 0, {eNS, 3}];
zG[d_] := L0 + m0 + (L01 + m01) d -
   ArcTan[Tan[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 + L01 d] Cos[Eps]];
(* Die Sprünge des Arcustangens durch Stetigmachen
  resp. Zusammensetzen überwinden:*)
updown[d_] := (1 - Sign[(zG[d] - 1)]) / 2 zG[d];
upup[d_] := (1 + Sign[(zG[d] - 1)]) / 2 (zG[d] - Pi);
up[d_] := updown[d] + upup[d];
downall[d_] := (1 - Sign[(zG[d] + 1)]) / 2 (zG[d] + Pi);
upall[d_] := (1 + Sign[(zG[d] + 1)]) / 2 up[d];
glattZG[d_] := (upall[d] + downall[d]);
zgMinuten[d_] := glattZG[d] / (2 Pi) 24 60;
Plot[zgMinuten[d] , {d, -191, 356}];
```



■ Deklination in Abhängigkeit der Tage

■ Direkte Lösung mit der Keplergleichung

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*)  Remove["Global`*"]
(* Konstanten *)
m0 = (356.5399 - 360) Degree; e = 0.0167092;
L0 = (282.9399 - 360) Degree; Eps = 23.4393 Degree;
L01 = 0.0000470684 Degree; m01 = 0.9856002664 Degree;
(* Keplerfunktion *)
f[eE_, d_] := eE - e Sin[eE] - m0 - m01 d;
eE[d_] := eNS /. FindRoot[f[eNS, d] == 0, {eNS, 3}];
delta[d_] :=
  ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 + L01 d] Sin[Eps]];
Plot[delta[d] / (2 Pi) 360, {d, 0, 366}];
```

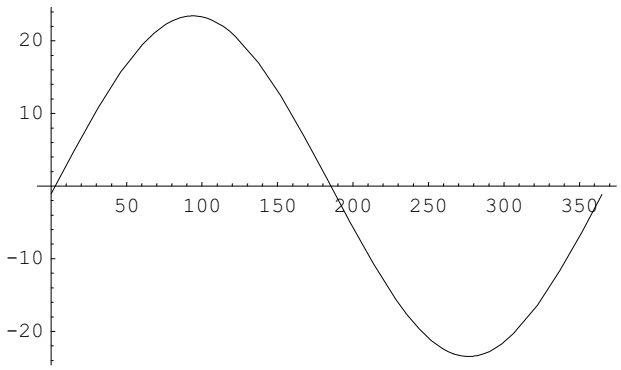


■ Näherungslösung

```
(* Aus Kalenderprogramm *)
m[n_] := 31; m[2] = 28; m[4] = 30; m[6] = 30;
m[9] = 30; m[11] = 30; m[11] = 30; m[14] = 28; m[-1] = 1;
(* Anzahl Tage seit 31.12. letzten Jahres *)
tage[tag_, mon_] := Sum[m[n], {n, 1, mon}] - m[mon] + tag;
day = 21; month = 12; d0 = tage[day, month] ;  Print[{day, month, d0}];
(*Deklination seit 21.3.,d.h.Frühlingsbeginn*)
delta1[d_] := ArcSin[Sin[m0 + L0 + (L01 + m01) (d)] Sin[Eps]] / (2 Pi) 360;
(*Anzahl Tage seit Frühlingsbeginn*)
 fpTage[tag_, mon_] := tage[tag, mon] - tage[21, 3];
day = 21; month = 3; d1 = fpTage[day, month]; Print[{day, month, d1}];
Plot[delta1[d + tage[20, 3]], {d, 0, 365}];
(* Beispiel Deklination am 29.8. *)
 Print[{"Datum ", 21, 6, " Deklination ", delta[tage[21, 6]] / (2 Pi) 360}];
 Print[{"Datum ", 29, 8, " Deklination ", delta[tage[29, 8]] / (2 Pi) 360}];
```

```
{21, 12, 355}
```

```
{21, 3, 0}
```



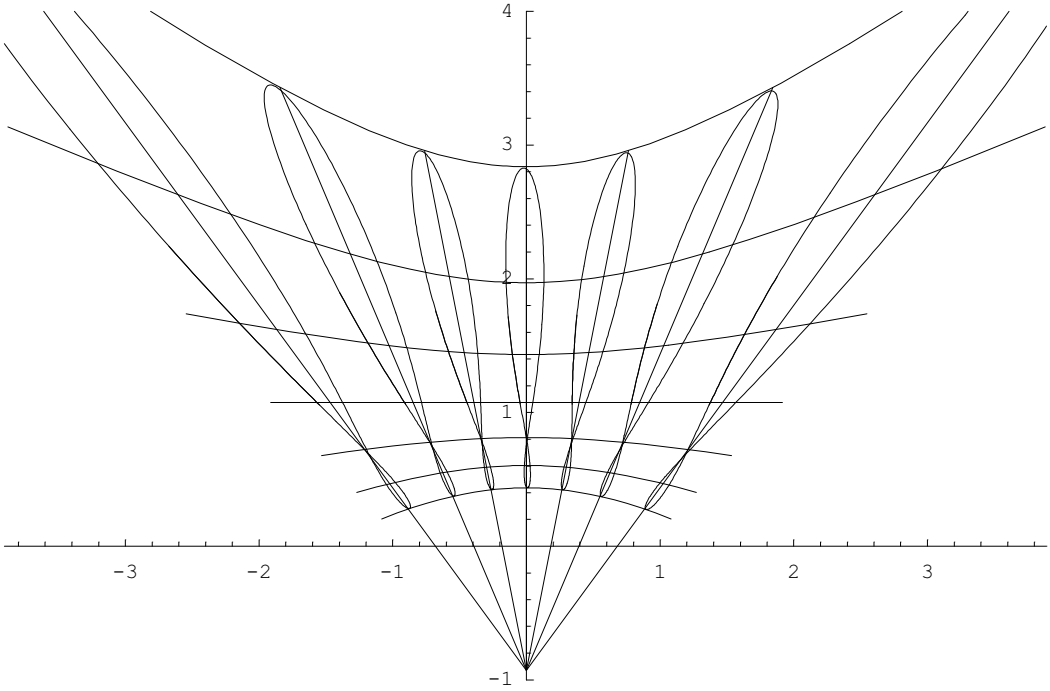

```
{Datum , 21, 6,  Deklination , 23.4382}
```

```
{Datum , 29, 8,  Deklination , 9.49856}
```

■ Horizontale Sonnenuhr für mittlere Zeit, Schlaufen für mittlere Zeit

■ Achtung: Braucht starke Maschine!

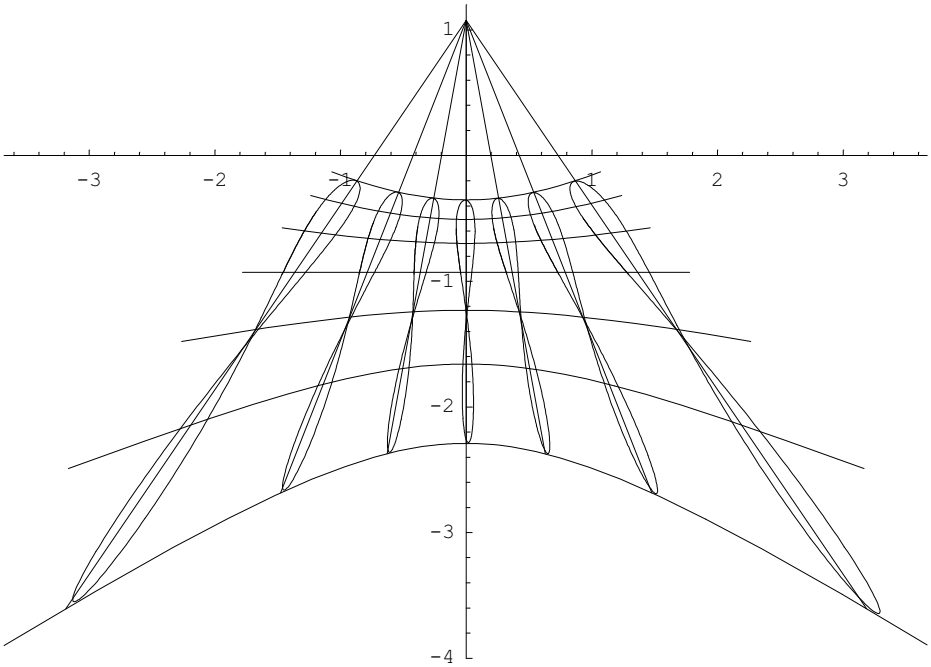
```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*)  Remove["Global`*"]
(* Konstanten *)
(* ===> Input, Altgrad *)  φ = 47.09; fi = (90 - Abs[φ]);
dek[1] = 23.5; dek[2] = 16 ; dek[3] = 8 ; dek[4] = 0 ;
dek[5] = -8 ; dek[6] = -16 ; dek[7] = -23.5;  g = 1;
hour = 2 Pi / 24;
m0 = (356.5399 - 360) Degree; e = 0.0167092;
L0 = (282.9399 - 360) Degree; Eps = 23.4393 Degree;
L01 = 0.0000470684 Degree; m01 = 0.9856002664 Degree;
(* Keplerfunktion *)
f[eE_, d_] := eE - e Sin[eE] - m0 - m01 d;
eE[d_] := eNS /. FindRoot[f[eNS, d] == 0, {eNS, 3}];
(* Zeitgleichung *) zG[d_] := L0 + m0 + (L01 + m01) d -
   ArcTan[Tan[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 + L01 d] Cos[Eps]];
(* Die Sprünge des Arcustangens durch Stetigmachen
  resp. Zusammensetzen überwinden:*)
updown[d_] := (1 - Sign[(zG[d] - 1)]) / 2 zG[d];
upup[d_] := (1 + Sign[(zG[d] - 1)]) / 2 (zG[d] - Pi);
up[d_] := updown[d] + upup[d];
downall[d_] := (1 - Sign[(zG[d] + 1)]) / 2 (zG[d] + Pi);
upall[d_] := (1 + Sign[(zG[d] + 1)]) / 2 up[d];
glattZG[d_] := (upall[d] + downall[d]);
zgStd[d_] := glattZG[d] / (2 Pi) 24 ;
(* Gnomonische Projektion *)
v[d_, fi_, t_] := g / (Cos[Sign[φ] d Degree] Cos[t hour] Sin[fi Degree] +
      Sin[Sign[φ] d Degree] Cos[fi Degree]) {(Cos[Sign[φ] d Degree] Sin[t hour]),
    Sign[φ] ((Cos[Sign[φ] d Degree] Cos[t hour] Cos[fi Degree] -
        Sin[Sign[φ] d Degree] Sin[fi Degree]) )};
(* Ursprung versetzt, vgl. Gnomonstab parallel Erdachse! *)
ursprung[φ_] := { 0, -g / Tan[φ Degree]};
(* Plot *) parPlot = ParametricPlot[Evaluate[Table[v[dek[k], fi, t], {k, 1, 7}]],
   {t, -3.5, 3.5}, AspectRatio → Automatic, PlotRange → {-1, 4},
   Epilog -> {Table[Line[{v[dek[7], fi, k], ursprung[φ]}], {k, -3, 3, 1}]},
   DisplayFunction → Identity];
tabd[n_] := Table[
   v[ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 + L01 d] Sin[Eps]] /
     Degree, fi, (n - zgStd[d])], {d, -100, 300}];
listp[n_] := ListPlot[tabd[n], PlotJoined → True, AspectRatio → Automatic,
   DisplayFunction → Identity];
listp0 = Show[listp[3], listp[2], listp[1], listp[0], listp[-1],
   listp[-2], listp[-3], DisplayFunction → Identity];
Show[parPlot, listp0, DisplayFunction → $DisplayFunction];
```


4
3
2
1
-1
-3
-2
-1
1
2
3

■ Vertikale Sonnenuhr für mittlere Zeit, Schlaufen für mittlere Zeit

■ Achtung: Braucht starke Maschine!

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*)  Remove["Global`*"]
(* Konstanten *)
(* ===> Input, Altgrad *)  φ = 47.09 - 90; fi = (90 - Abs[φ]);
dek[1] = 23.5; dek[2] = 16 ; dek[3] = 8 ; dek[4] = 0 ;
dek[5] = -8 ; dek[6] = -16 ; dek[7] = -23.5;  g = 1;
hour = 2 Pi / 24;
m0 = (356.5399 - 360) Degree; e = 0.0167092;
L0 = (282.9399 - 360) Degree; Eps = 23.4393 Degree;
L01 = 0.0000470684 Degree; m01 = 0.9856002664 Degree;
(* Keplerfunktion *)
f[eE_, d_] := eE - e Sin[eE] - m0 - m01 d;
eE[d_] := eNS /. FindRoot[f[eNS, d] == 0, {eNS, 3}];
(* Zeitgleichung *)zG[d_] := L0 + m0 + (L01 + m01) d -
   ArcTan[Tan[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 + L01 d] Cos[Eps]];
(* Die Sprünge des Arcustangens durch Stetigmachen
  resp. Zusammensetzen überwinden:*)
updown[d_] := (1 - Sign[(zG[d] - 1)]) / 2 zG[d];
upup[d_] := (1 + Sign[(zG[d] - 1)]) / 2 (zG[d] - Pi);
up[d_] := updown[d] + upup[d];
downall[d_] := (1 - Sign[(zG[d] + 1)]) / 2 (zG[d] + Pi);
upall[d_] := (1 + Sign[(zG[d] + 1)]) / 2 up[d];
glattZG[d_] := (upall[d] + downall[d]);
zgStd[d_] := glattZG[d] / (2 Pi) 24 ;
(* Gnomonische Projektion *)
v[d_, fi_, t_] := g / (Cos[Sign[φ] d Degree] Cos[t hour] Sin[fi Degree] +
      Sin[Sign[φ] d Degree] Cos[fi Degree]) {(Cos[Sign[φ] d Degree] Sin[t hour]),
    Sign[φ] ((Cos[Sign[φ] d Degree] Cos[t hour] Cos[fi Degree] -
        Sin[Sign[φ] d Degree] Sin[fi Degree]) )};
(* Ursprung versetzt, vgl. Gnomonstab parallel Erdachse! *)
ursprung[φ_] := { 0, -g / Tan[φ Degree]};
(* Plot *)
parPlot = ParametricPlot[Evaluate[Table[v[dek[k], fi, t], {k, 1, 7}]], {t, -3.5, 3.5},
   AspectRatio → Automatic, PlotRange → {-4, 1.2},
   Epilog -> {Table[Line[{ v[dek[1], fi, k], ursprung[φ]}], {k, -3, 3, 1}]},
   DisplayFunction → Identity];
tabd[n_] := Table[
   v[ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 + L01 d] Sin[Eps]] /
     Degree, fi, (n - zgStd[d])], {d, -100, 300}];
listp[n_] := ListPlot[tabd[n], PlotJoined → True, AspectRatio → Automatic,
   DisplayFunction → Identity];
listp0 = Show[listp[3], listp[2], listp[1], listp[0], listp[-1],
   listp[-2], listp[-3], DisplayFunction → Identity];
Show[parPlot, listp0, DisplayFunction → $DisplayFunction];
```



Vorschlag für Module

- **Putz: Alle Daten löschen, Neubeginn**

- **Programm (* INITIALISATION *)**

```
(*Old Form:Remove["Global`@*"]*)
putz = Remove["Global`*"];(* "Putzmaschine" *)
```

- **dataEntry:**
  Stellt notwendige Parameter und globale Variablen zur Verfügung

- **Programm (* DEFINITION und INITIALISATION *)**

```
(* Standartwerte φ von Biel in Altgrad , g = 1 *)

dataEntryFixData = Module[{},
   m0 = (356.5399 - 360) Degree;
   e = 0.0167092;
   L0 = (282.9399 - 360) Degree;
   Eps = 23.4393 Degree;
   L01 = 0.0000470684 Degree;
   m01 = 0.9856002664 Degree;
   Print["OUT-Variabeln: m0, e, L0, Eps, L01, m01"];];
```

```
OUT-Variabeln: m0, e, L0, Eps, L01, m01
```

■ Beispiel

```
m0
```

```
-0.0603901
```

■ Programm (* DEFINITION*)

```
dataEntry[TAKEφ_, TAKEg_, jahr_] := Module[{ φ0 = 47.09, g0 = 1, intYeahr},
   dataEntryFixData;
   (* MP berechnen,Formel:*)
   φ = φ0 (1 - Sign[TAKEφ]) (1 + Sign[TAKEφ]) + TAKEφ;
   Print["φ = ", φ];
   φVert = -Sign[φ] * (90 - Abs[φ]);
   g = g0 (1 - Sign[TAKEg]) (1 + Sign[TAKEg]) + TAKEg;
   Print["g = ", g];
   (* Stunden ===> Rad-Umrechnungsfaktor *)
   hour = 2 Pi / 24 ;
   (* Jahrestage rechnen *)
   jahresTage[intYeahr_] :=
    365 Mod[intYeahr, 2000] + Floor[Mod[ Mod[intYeahr, 2000], 2000] / 4];
   jahresTage[jahr];
   jahrOut = jahr;
   Print["OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut"]; ];
dataEntry[0, 0, 0]
```

```
φ = 47.09

g = 1

OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut
```

■ Beispiele (* INITIALISATION *)

```
dataEntry[30, 50, 12]
```

```
φ = 30

g = 50

OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut
```

```
dataEntry[0, 0, 0]; φVert
```

```
φ = 47.09

g = 1

OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut

-42.91
```

```
{g, φ}
```

```
{1, 47.09}
```

```
φVert
```

```
-42.91
```

```
{jahresTage[12], jahresTage[2012]}
```

```
{4383, 4383}
```

```
dataEntry[30, 50, 01]; jahrOut
```

```
φ = 30
```

```
g = 50
```

```
OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut
```

```
1
```

- **ombrixGzentr, ombrixGzentrDeltaStd:**
  Zeichnet Ombrix-Zifferblatt für eingegebene Stunden, benutzt dataEntry.
  Benutzte Module müssen allgemein initialisiert sein!

- **Programm (* DEFINITION*)**

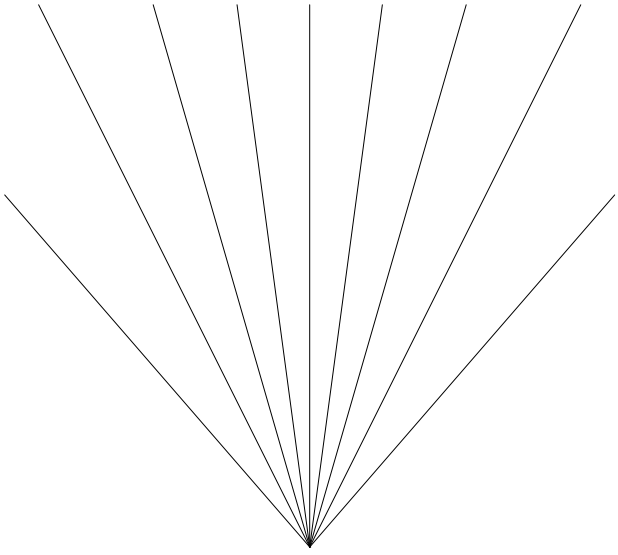
```
(* ===> INPUT, hMin= 1. Stunde, hMax=Letzte Stunde (h) *)
(* Beispiel: Werte hMin= -5 h,-4 h,...,0 h,..., hMax=5 h *)

ombrixGzentr[hMin_, hMax_] := Module[{tVar, φVar},
   Print["Der Modul dataEntry wird hier vorausgesetzt!"];
   (* Ost-West-Koordinate:*)
   MP[tVar_, φVar_] := g Tan[tVar hour] / Sin[(90 - φVar) Degree];
   (* Ursprung versetzt, Nord-Süd-Koordinate *)
   ursprung[φVar_] := {0, -g / Tan[φVar Degree]};
   (* Tabelle der Plotpunkte rechnen *)
   tabPlotPunkte[φVar_] :=
    Table[{MP[tVar, φVar], g / Tan[(90 - φVar) Degree]}, {tVar, hMin, hMax, 1}];
   Print["tabPlotPunkte = ", tabPlotPunkte[φ] // N];
   (* OmbrixPlot machen ===> Ombrix-Zifferblatt *)
   ombixPlot = Show[Graphics[Table[Line[{ursprung[φ], tabPlotPunkte[φ][[n]]}],
       {n, 1, Length[tabPlotPunkte[φ]]}]], AspectRatio → Automatic];
   Print["OUT-Funktionen: MP, ursprung, tabPlotPunkte, ombixPlot"];];
```

```
(* ===> INPUT, hMin= 1. Stunde, hMax=Letzte Stunde (h) *)
(* Beispiel: Werte hMin= -5 h,-4 h,...,0 h,..., hMax=5 h *)

ombrixGzentrDeltaStd[hMin_, hMax_, deltaStd_] := Module[{tVar, φVar, diff},
   Print["Der Modul dataEntry wird hier vorausgesetzt!"];
   (* Ost-West-Koordinate:*)
   MP[tVar_, φVar_, diff_] := g Tan[(tVar - diff) hour] / Sin[(90 - φVar) Degree];
   (* Ursprung versetzt, Nord-Süd-Koordinate *)
   ursprung[φVar_] := {0, -g / Tan[φVar Degree]};
   (* Tabelle der Plotpunkte rechnen *)
   tabPlotPunkte[φVar_] := Table[
     {MP[tVar, φVar, deltaStd], g / Tan[(90 - φVar) Degree]}, {tVar, hMin, hMax, 1}];
   Print["tabPlotPunkte = ", tabPlotPunkte[φ] // N];
   (* OmbrixPlot machen ===> Ombrix-Zifferblatt *)
   ombixPlot = Show[Graphics[Table[Line[{ursprung[φ], tabPlotPunkte[φ][[n]]}],
       {n, 1, Length[tabPlotPunkte[φ]]}]], AspectRatio → Automatic];
   Print["OUT-Funktionen: MP, ursprung, tabPlotPunkte, ombixPlot"];];
```

■ Beispiele (* INITIALISATION *)

```
dataEntry[0, 0]; ombrixGzentr[-4, 4]
```

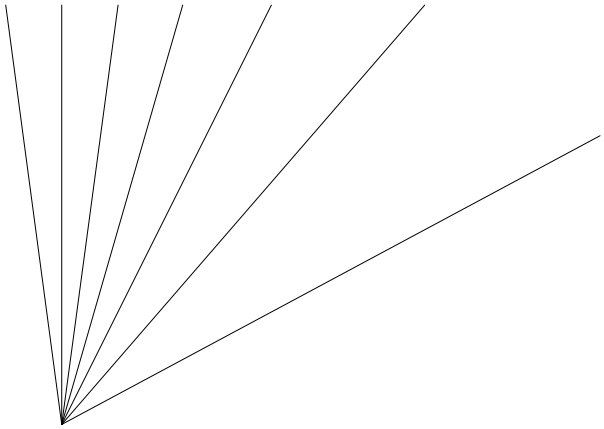
```
Der Modul dataEntry wird hier vorausgesetzt!

tabPlotPunkte = {{-100., 28.8675}, {-57.735, 28.8675}, {-33.3333, 28.8675}, {-15.4701, 28.8675},
  {0., 28.8675}, {15.4701, 28.8675}, {33.3333, 28.8675}, {57.735, 28.8675}, {100., 28.8675}}
```



```
OUT-Funktionen: MP, ursprung, tabPlotPunkte, ombixPlot
```

dataEntry[0, 0]; ombrixGzentr[-1, 5]

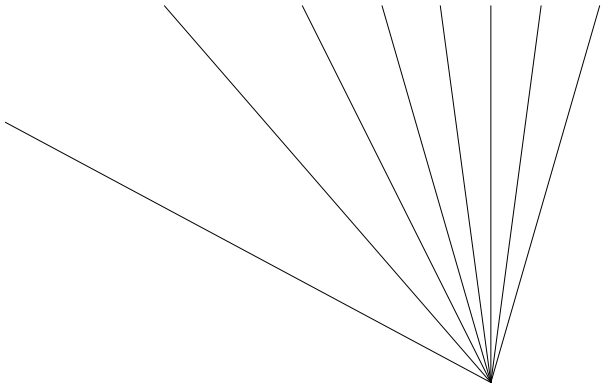
```
Der Modul dataEntry wird hier vorausgesetzt!

tabPlotPunkte = {{-15.4701, 28.8675}, {0., 28.8675}, {15.4701, 28.8675},
  {33.3333, 28.8675}, {57.735, 28.8675}, {100., 28.8675}, {215.47, 28.8675}}
```



```
OUT-Funktionen: MP, ursprung, tabPlotPunkte, ombixPlot
```

ombrixGzentrDeltaStd[-3, 4, 2]

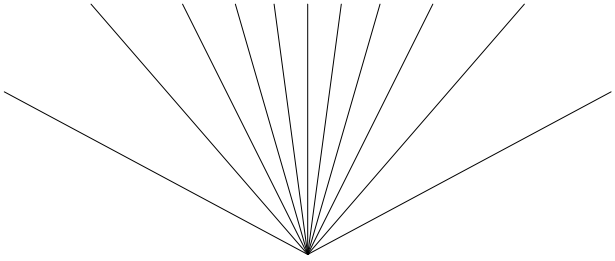
```
Der Modul dataEntry wird hier vorausgesetzt!

tabPlotPunkte = {{-215.47, 28.8675}, {-100., 28.8675}, {-57.735, 28.8675}, {-33.3333, 28.8675},
  {-15.4701, 28.8675}, {0., 28.8675}, {15.4701, 28.8675}, {33.3333, 28.8675}}
```



```
OUT-Funktionen: MP, ursprung, tabPlotPunkte, ombixPlot
```

dataEntry[φ - 90, 1]; ombrixGzentr[-5, 5]

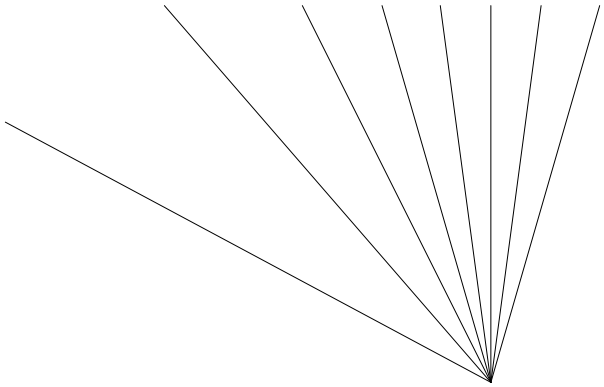
```
Der Modul dataEntry wird hier vorausgesetzt!

tabPlotPunkte = {{-215.47, 28.8675}, {-100., 28.8675}, {-57.735, 28.8675},
  {-33.3333, 28.8675}, {-15.4701, 28.8675}, {0., 28.8675}, {15.4701, 28.8675},
  {33.3333, 28.8675}, {57.735, 28.8675}, {100., 28.8675}, {215.47, 28.8675}}
```



```
OUT-Funktionen: MP, ursprung, tabPlotPunkte, ombixPlot
```

ombrixGzentrDeltaStd[-3, 4, 2]

```
Der Modul dataEntry wird hier vorausgesetzt!

tabPlotPunkte = {{-215.47, 28.8675}, {-100., 28.8675}, {-57.735, 28.8675}, {-33.3333, 28.8675},
  {-15.4701, 28.8675}, {0., 28.8675}, {15.4701, 28.8675}, {33.3333, 28.8675}}
```



```
OUT-Funktionen: MP, ursprung, tabPlotPunkte, ombixPlot
```

■ Kalendermodul:
Berechnet Anzahl Tage des eingegebenen Datums ab Jahresbeginn oder Frühlingsbeginn

■ Programm (* DEFINITION*)

```
(* tage ===> Berechnung der Anzahl Tage seit 31.12. letzten Jahres *)
(* fpTage ===> Berechnung der Anzahl Tage seit letztem Frühlingsbeginn,
 Jan. = 13. Mt., Feb. = 14. Mt. *)
(* tageS, fpTageS ===> für Schaltjahre, ein Tag mehr (ein Tag mehr) *)

kalender[jour_, mois_] := Module[{m, n},
  m[n_] := 31; m[2] = 28; m[4] = 30; m[6] = 30; m[9] = 30; m[11] = 30; m[14] = 28; m[0] = 0;
  tage[day_, mon_] = Sum[m[n], {n, 1, mon}] - m[mon] + day;
  tageSJ[day_, mon_] = If[mon < 3, tage[day, mon], tage[day, mon] + 1];
  fpTage[day_, mon_] = tage[day, mon] - tage[21, 3];
  fpTageSJ[day_, mon_] = tage[day, mon] - tage[21, 3] + 1;
  Print["   tage[jour,mois] = ", tage[jour, mois],
   "   tageSJ[jour,mois] = ", tageSJ[jour, mois],
   "   fpTage[jour,mois] = ", fpTage[jour, mois],
   "   fpTageSJ[jour,mois] = ", fpTageSJ[jour, mois] ];
  Print["OUT-Funktionen: tage ,tageSJ, fpTage, fpTageSJ"];
  Print["Funktioniert für positiv eingegebene Werte"];]
```

■ Beispiele (* INITIALISATION *)

```
kalender[21, 3]
```

```
   tage[jour,mois] = 83   tageSJ[jour,mois] =
 84   fpTage[jour,mois] = 0   fpTageSJ[jour,mois] = 1

OUT-Funktionen: tage ,tageSJ, fpTage, fpTageSJ

Funktioniert für positiv eingegebene Werte
```

```
{tage[21, 3] , tage[6, 5], tage[1, 1]}
```

```
{83, 130, 1}
```

```
{tage[31, 0], tage[30, 0], tage[1, 0], tage[0, 0]}
```

```
{0, -1, -30, -31}
```

```
{{"Test", "21.3.", kalender[21, 3]},
 {"Test", "29.8.", kalender[29, 8]},
 {"Test", "21.12.", kalender[21, 12]}, {"Test", "31.12.", kalender[20, 15]},
 {"Test", "21.3.", tage[21, 3]},
 {"Test", "29.8.", tage[29, 8]},
 {"Test", "21.12.", tage[21, 12]},
 {"Test", "31.12.", tage[31, 12]}}
```

```
   tage[jour,mois] = 83   tageSJ[jour,mois] =
 84   fpTage[jour,mois] = 0   fpTageSJ[jour,mois] = 1

OUT-Funktionen: tage ,tageSJ, fpTage, fpTageSJ

Funktioniert für positiv eingegebene Werte

   tage[jour,mois] = 246   tageSJ[jour,mois] =
 247   fpTage[jour,mois] = 163   fpTageSJ[jour,mois] = 164

OUT-Funktionen: tage ,tageSJ, fpTage, fpTageSJ

Funktioniert für positiv eingegebene Werte

   tage[jour,mois] = 362   tageSJ[jour,mois] =
 363   fpTage[jour,mois] = 279   fpTageSJ[jour,mois] = 280

OUT-Funktionen: tage ,tageSJ, fpTage, fpTageSJ

Funktioniert für positiv eingegebene Werte

   tage[jour,mois] = 454   tageSJ[jour,mois] =
 455   fpTage[jour,mois] = 371   fpTageSJ[jour,mois] = 372

OUT-Funktionen: tage ,tageSJ, fpTage, fpTageSJ

Funktioniert für positiv eingegebene Werte

{{Test, 21.3., Null}, {Test, 29.8., Null}, {Test, 21.12., Null}, {Test, 31.12., Null},
 {Test, 21.3., 83}, {Test, 29.8., 246}, {Test, 21.12., 362}, {Test, 31.12., 372}}
```

■ zeitGleichung:
Berechnet Zeitgleichung und stellt Funktionen zur Verfügung:

■ Programm (* DEFINITION*)

```
zeitGleichung[plotYes1_, jahr_] := Module[{disp},
  (* putz; *)
  (* Jahr auf 2 Stellen angeben. Z.B. 00 statt 2000 *)
  dataEntry[0, 0, jahr];
  (* Keplerfunktion *)
  f[eE_, d_] := eE - e Sin[eE] - m0 - m01 jahresTage[jahr] - m01  d;
  eE[d_] := eNS /. FindRoot[f[eNS, d] == 0, {eNS, 3}];
  zG[d_] := L0 + m0 + (L01 + m01) d  + Mod[(L01 + m01) jahresTage[jahr],
     360] - ArcTan[Tan[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] +
        L0 + L01 d + L01 jahresTage[jahr]] Cos[Eps]] - jahrOut 2 Pi;
  (* Die Sprünge des Arcustangens durch Stetigmachen
    resp. Zusammensetzen überwinden:*)
  updown[d_] := (1 - Sign[(zG[d] - 1)]) / 2 zG[d];
  upup[d_] := (1 + Sign[(zG[d] - 1)]) / 2 (zG[d] - Pi);
  up[d_] := updown[d] + upup[d];
  downall[d_] := (1 - Sign[(zG[d] + 1)]) / 2 (zG[d] + Pi);
  upall[d_] := (1 + Sign[(zG[d] + 1)]) / 2 up[d];
  glattZG[d_] := (upall[d] + downall[d]);
  zgStd[d_] := glattZG[d] / (2 Pi) 24 ;
  zgMinuten[d_] := glattZG[d] / (2 Pi) 24 60;
  If[plotYes1 == 1, Goto[erstens], Goto[zweitens]];
  Label[erstens]; Plot[zgMinuten[day] , {day, -191, 356}]; Goto[drittens];
  Label[zweitens]; Print["Kein Plot, nur Initialisierung"];
  Label[drittens];
  Print["OUT-Funktionen: f, eE, zG, updown,
     upup, up, downall, upall, glattZG, zgStd, zgMinuten"];
  Print["Jahr auf 2 Stellen angeben. Z.B. 00 statt 2000"];]
```

■ Beispiele (* INITIALISATION *)

```
? eE
```

```
Global`eE
```

```
zeitGleichung[2, 01]
```

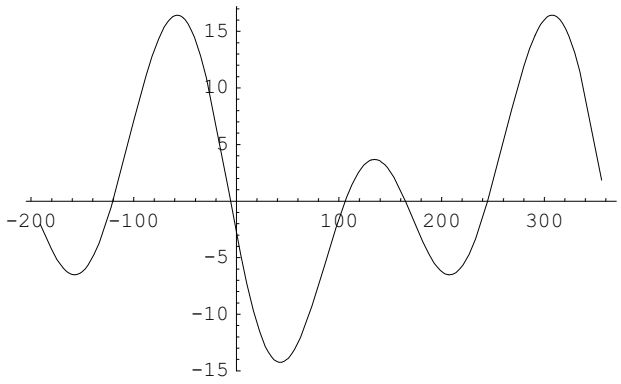
```
φ = 47.09

g = 1

OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut

Kein Plot, nur Initialisierung

OUT-Funktionen: f, eE, zG, updown, upup, up, downall, upall, glattZG, zgStd, zgMinuten

Jahr auf 2 Stellen angeben. Z.B. 00 statt 2000
```

zeitGleichung[1, 01]

φ = 47.09

g = 1

OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut



OUT-Funktionen: f, eE, zG, updown, upup, up, downall, upall, glattZG, zgStd, zgMinuten

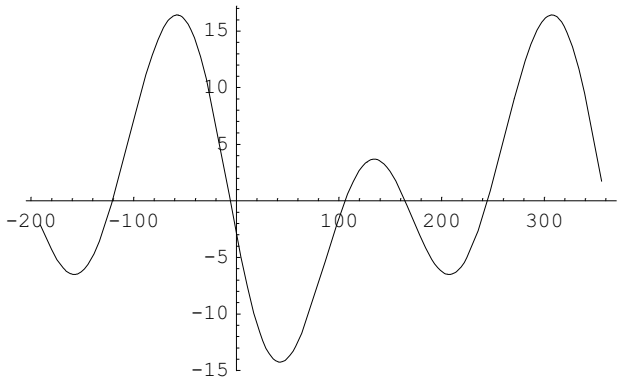
Jahr auf 2 Stellen angeben. Z.B. 00 statt 2000

zeitGleichung[1, 00]

φ = 47.09

g = 1

OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut



OUT-Funktionen: f, eE, zG, updown, upup, up, downall, upall, glattZG, zgStd, zgMinuten

Jahr auf 2 Stellen angeben. Z.B. 00 statt 2000

```
zeitGleichung[2, 01]; zgStd[5]
```

```
φ = 47.09

g = 1

OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut

Kein Plot, nur Initialisierung

OUT-Funktionen: f, eE, zG, updown, upup, up, downall, upall, glattZG, zgStd, zgMinuten

Jahr auf 2 Stellen angeben. Z.B. 00 statt 2000

-0.0840585
```

■ deklin:
Deklinationsbestimmung zum Datum in Altgrad, setzt dataEntry, zeitGleichung, kalender (und tage) voraus. Benutzte Module müssen allgemein initialisiert sein!

■ Programm (* DEFINITION*)

```
deklin[myTag_, myMonat_] := Module[{dayNow},
   dayNow = tage[myTag, myMonat];
   deltaAngle[dd_] := 360 / (2 Pi)
     ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[Evaluate[eE[dd]] / 2 ]] + L0 + L01 dd]
       Sin[Eps]];
   Print["OUT-Funktionen: deltaAngle = ", deltaAngle[tage[myTag, myMonat]] ];
   deltaAngle[dayNow] ];
```

■ Beispiele (* INITIALISATION *)

```
(* Initialisierung *)
dataEntry[0, 0]; zeitGleichung[0, 01]; kalender[29, 8]; tage[29, 8]; deklin[0, 0]
```

```
φ = 47.09

g = 1

OUT-Variabeln: φ, φVert, g, hour, jahresTage, jahrOut

Kein Plot, nur Initialisierung

OUT-Funktionen: f, eE, zG, updown, upup, up, downall, upall, glattZG, zgStd, zgMinuten

Jahr auf 2 Stellen angeben. Z.B. 00 statt 2000

   tage[jour,mois] = 246   tageSJ[jour,mois] =
 247   fpTage[jour,mois] = 163   fpTageSJ[jour,mois] = 164

OUT-Funktionen: tage ,tageSJ, fpTage, fpTageSJ

Funktioniert für positiv eingegebene Werte

OUT-Funktionen: deltaAngle = -21.5732

-21.5732
```

```
tage[1, 1]
```

```
1
```

```
deltaAngle[0]
```

```
-23.1272
```

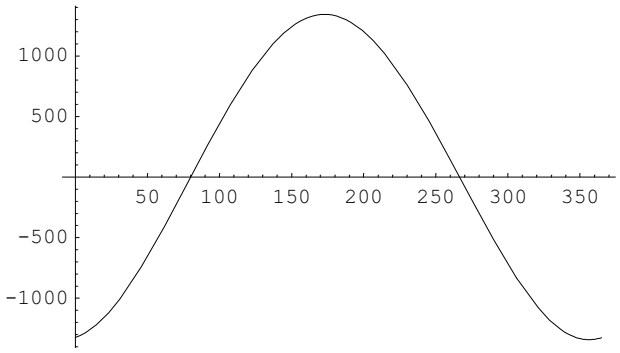
```
(* Kontrollen *)  ?eE
```

```
Global`eE

eE[d_] := eNS /. FindRoot[f[eNS, d] == 0, {eNS, 3}]
```

```
e
```

```
0.0167092
```

```
Plot[deltaAngle[dd] / (2 Pi) 360, {dd, 0, 365}];
```




```
(* Test *)  {deklin[28, 8], deklin[21, 12], deklin[17.805, 3], deklin[21, 3]}
```

```
OUT-Funktionen: deltaAngle = 8.15565

OUT-Funktionen: deltaAngle = -23.3125

OUT-Funktionen: deltaAngle = -0.102542

OUT-Funktionen: deltaAngle = 1.15866

{8.15565, -23.3125, -0.102542, 1.15866}
```

- **dekPlot:**
  Plot von n Deklinationslinien, setzt deklin (und somit dataEntry, zeitGleichung, kalender (und tage)) voraus.
  Benutzte Module müssen allgemein initialisiert sein!

- **Programm (* DEFINITION*)**

```
dekPlot[dataListe_, horVert_, tMin_,
   tMax_, yRangeMin_, yRangeMax_, plotYes1_] := Module[{fi},
    If[horVert == 1, fi = φ, If[horVert == 0, fi = φVert,
     Print["Vertikal: horVert=0 und horizontal horVert = 1 !!!!"]]];
   (* Definitionen und Rechnungen *)
   v[d_, fi_, t_] := g / (Cos[Sign[φ] d Degree] Cos[t hour] Sin[fi Degree] +
        Sin[Sign[φ] d Degree] Cos[fi Degree])
     {(Cos[Sign[φ] d Degree] Sin[t hour]), Sign[φ] ((Cos[Sign[φ] d Degree]
          Cos[t hour] Cos[fi Degree] - Sin[Sign[φ] d Degree] Sin[fi Degree]) )};
   If[plotYes1 == 1, Goto[erstens], Goto[zweitens]];
   Label[erstens]; ParametricPlot[
    Evaluate[Table[v[dataListe[[j]], fi, t], {j, 1, Length[dataListe]}]],
    {t, tMin, tMax}, AspectRatio → Automatic, PlotRange → {yRangeMin, yRangeMax}];
   Print["Goto ist verwendet worden"];
   Goto[drittens];
   Label[zweitens]; Print["Kein Plot, nur Initialisierung von v[d,fi,t]!"];
   Print["Goto ist verwendet worden"];
   Label[drittens];
   Print["OUT-Funktionen: v"]; ];
```

- **Beispiel (* INITIALISATION *)**

```
(* ===> Input von Hand *)
n = 8; d[1] = deklin[20, 12]; d[2] = deklin[1, 2];
d[3] = deklin[1, 3]; d[4] = deklin[17.805, 3]; d[5] = deklin[1, 5];
d[6] = deklin[1, 6]; d[7] = deklin[21, 6]; d[8] = deklin[29, 8];
(* ===> Listengenerierung mit dem Input *)
dlist[n_] := Table[d[k], {k, 1, n}];
dlist[n]
```

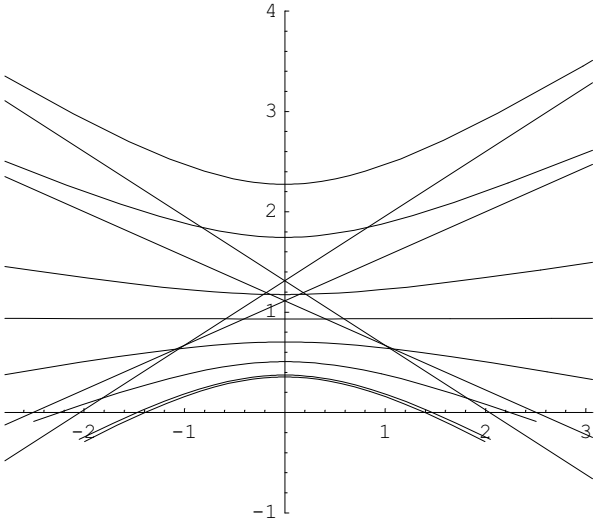
```
OUT-Funktionen: deltaAngle = -23.3532

OUT-Funktionen: deltaAngle = -17.2842

OUT-Funktionen: deltaAngle = -6.69083

OUT-Funktionen: deltaAngle = -0.102542

OUT-Funktionen: deltaAngle = 16.0643

OUT-Funktionen: deltaAngle = 22.4714

OUT-Funktionen: deltaAngle = 23.4045

OUT-Funktionen: deltaAngle = 7.79124

{-23.3532, -17.2842, -6.69083, -0.102542, 16.0643, 22.4714, 23.4045, 7.79124}
```

(* ===> Plot der Liste *)

dekPlot[dlist[n], 1, -5, 5, -1, 4, 0]

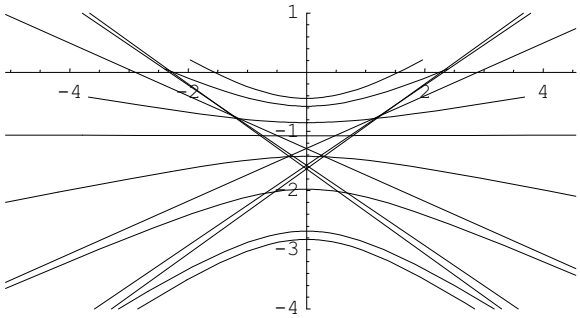
```
Kein Plot, nur Initialisierung von v[d,fi,t]!

Goto ist verwendet worden

OUT-Funktionen: v
```

dekPlot[dlist[n], 1, -5, 5, -1, 4, 1]




```
Goto ist verwendet worden

OUT-Funktionen: v
```

dekPlot[dlist[n], 0, -5, 5, -4, 1, 1]




```
Goto ist verwendet worden

OUT-Funktionen: v
```

■ zeitSchlaufen

Plot von n Zeitschlaufen für Stunden, setzt dekPlot, deklin (und somit dataEntry, zeitGleichung, kalender (und tage)) voraus.
Benutzte Module müssen allgemein initialisiert sein!

■ Programm (* DEFINITION*)

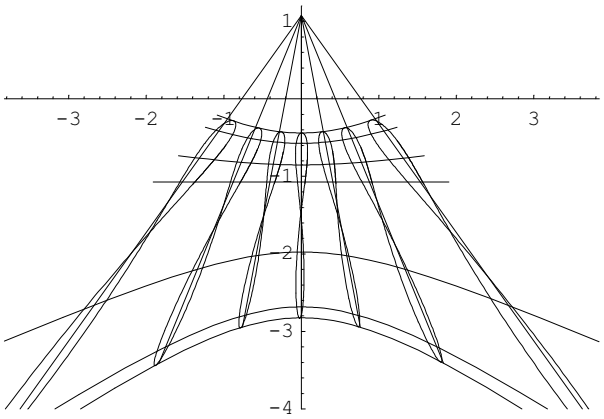
```
zeitSchlaufen[dataListe_, horVert_, maxDekNummer_, {paramMin_, paramMax_},
   {rangeMin_, rangeMax_}, {minNumber_, maxNumber_}] := Module[{fi},
    If[horVert == 1, fi = φ, If[horVert == 0, fi = φVert,
     Print["Vertikal: horVert=0 und horizontal horVert = 1 !!!!"];
     Goto[endeProgramm]]];
   parPlot = ParametricPlot[Evaluate[Table[v[dataListe[[k]], fi, t],
       {k, 1, Length[dataListe]}]], {t, paramMin, paramMax},
     AspectRatio → Automatic, PlotRange → {rangeMin, rangeMax},
     Epilog -> {Table[Line[{ v[dataListe[[maxDekNummer]], fi, k], ursprung[fi]}],
        {k, minNumber, maxNumber, 1}]}, DisplayFunction → Identity];
   tabd[n_] := Table[v[ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] +
          L0 + L01 d] Sin[Eps]] / Degree, fi, (n - zgStd[d])], {d, -100, 300}];
   listp[n_] := ListPlot[tabd[n], PlotJoined → True, AspectRatio → Automatic,
     DisplayFunction → Identity];
   listp0 = Show[Table[listp[j], {j, minNumber, maxNumber, 1}],
     DisplayFunction → Identity];
   Show[parPlot, listp0, DisplayFunction → $DisplayFunction];
   Label[endeProgramm];
   Print["Ende"];
   Print["OUT-Funktionen und Variablen: parPlot, tabd, listp, listp0"];];
(* Alternativdefinition ohne Klammern *)
zeitSchlaufen[dataListe_, horVert_, maxDekNummer_,
  paramMin_, paramMax_, rangeMin_, rangeMax_, minNumber_, maxNumber_] :=
 zeitSchlaufen[dataListe, horVert, maxDekNummer, {paramMin, paramMax},
  {rangeMin, rangeMax}, {minNumber, maxNumber}]
```

■ Beispiele (* INITIALISATION *)

```
zeitSchlaufen[dlist[7], 2, 7, {-3.5, 3.5}, {-4, 1.2}, {-3, 3}]
```

```
Vertikal: horVert=0 und horizontal horVert = 1 !!!!

Ende

OUT-Funktionen und Variablen: parPlot, tabd, listp, listp0
```

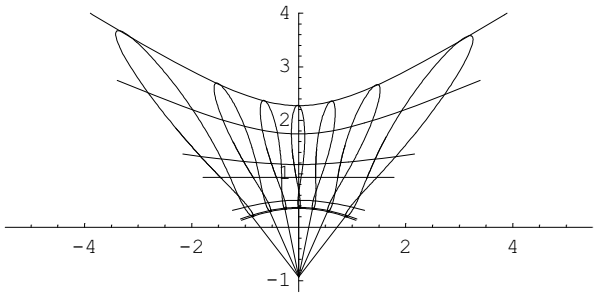
zeitSchlaufen[dlist[7], 0, 7, {-3.5, 3.5}, {-4, 1.2}, {-3, 3}]



Ende

OUT-Funktionen und Variablen: parPlot, tabd, listp, listp0

zeitSchlaufen[dlist[7], 1, 7, {-3.5, 3.5}, {-1.2, 4}, {-3, 3}]



Ende

OUT-Funktionen und Variablen: parPlot, tabd, listp, listp0

■ Drehungen, Wandtranslation in Horizontallage

■ Programm (* DEFINITION*)

```
newPositionWithPrint[ostNord_, zenitNord_, φ_] :=
  Module[{a, b, φOld, u, v, w, d, r, r1, R, dY, dZ, xKS, yKS, zKS, goAeq, xNew, yNew,
    zNew, solvU, solvV, uRichtung, vRichtung, u1, u2, u3, v1, v2, v3, vv},
   (* Umrechnung mit Vektorprodukt *)
   (* ===> INPUT, Altgrad *)
   a = ostNord ; b = zenitNord ; φOld = φ;
   u = {Cos[a Degree], Sin[a Degree], 0}; Print["u = ", u // N];
   v = {-Sin[a Degree] Sin[b Degree], Cos[a Degree] Sin[b Degree], Cos[b Degree]};
   Print["v = ", v // N];
   (* w: Normalenvektor im Ortshorizontsystem *)
   w = Transpose[{Cross[u, v]}]; Print["w = ", w // N // MatrixForm];
   (* Drehung um Ost-West-Achse im Ortshorizontsystem *)
   (* Drehung ins Orstäquatorsystem *)
   φOld = φ; d = (90 - φOld) ; Print["d = ", d // N];
   dreh = {{1, 0, 0}, {0, Cos[d Degree], -Sin[d Degree]},
     {0, Sin[d Degree], Cos[d Degree]}}; Print["dreh = " , dreh // N // MatrixForm];
   r = dreh.w; Print["r = ", r // N // MatrixForm];
   r1 = Flatten[r]; Print["r1 = ", r1 // N];
   R = {- r1[[2]], -r1[[1]], r1[[3]]}; Print["R = ", R // N];
   Delta1 = Sign[r1[[3]]]
     ArcCos[Sqrt[(R[[1]]^2 + R[[2]]^2) / (R[[1]]^2 + R[[2]]^2 + R[[3]]^2)]] ;
   (* Delta1 ist Winkel zwischen Vektor und Ortsäquatorebene,
    d.h. die neue geographische Breite *)
   Delta2 = Sign[r1[[3]]] ArcCos[Sqrt[R[[1]]^2 / (R[[1]]^2 + R[[2]]^2)]];
   (* Delta2 ist Winkel zwischen
     Vektorprojektion in Ortsäquatorebene x resp. Süd *)
   dY = Delta1; dZ = Delta2;
   Delta1Grad = Delta1 / Degree // N;
   Delta2Grad = Delta2 / (2 Pi) 360 // N;; Delta2Std = Delta2 / (2 Pi) 24 // N;
   Print["Delta1Grad = ", Delta1Grad, " Vorzeichen= " , Sign[r1[[3]]]];
Print["Delta2Grad = ", Delta2Grad,
    ", Delta2Std = ", Delta2Std, " Vorzeichen= " , Sign[r1[[2]]]];
   (*Umrechnen des Vertikalvektors der Wand zur Drehung
     des Outputs in Vertikallage, in Einzelschritten*)
   Print["Test Matrixprodukt ===> ", dreh.v == Flatten[dreh.Transpose[{v}]]];
   xKS = {1, 0, 0}; yKS = {0, 1, 0}; zKS = {0, 0, 1};
   Print["Koordinatensystem : ", {xKS, yKS, zKS} // MatrixForm ]
    goAeq[vec_] := {-(dreh.vec)[[2]], -(dreh.vec)[[1]], (dreh.vec)[[3]]};
   Print["Prozedur Koord.Umrechnung
      von - r1[[2]],-r1[[1]],r1[[3]] auf dreh.vec anwenden"];
   vAeq = goAeq[v]; uAeq = goAeq[u]; xAeq = goAeq[xKS];
   yAeq = goAeq[yKS]; zAeq = goAeq[zKS];
   Print["===>vAeq = ", vAeq // N];
   Print["===>uAeq = ", uAeq // N];
   Print["===>xAeq = ", xAeq // N];
   Print["===>yAeq = ", yAeq // N];
   Print["===>zAeq = ", zAeq // N];
   drehZ = {{Cos[dZ], -Sin[dZ], 0}, {Sin[dZ], Cos[dZ], 0}, {0, 0, 1}};
```

```
Print["drehZ =", drehZ // N // MatrixForm];
drehY = {{Cos[dY], 0, -Sin[dY]}, {0, 1, 0}, {Sin[dY], 0, Cos[dY]}};
Print["drehY =", drehY // N // MatrixForm];
goNew[vec_] := drehZ.(drehY.vec);
Print["Vektor um Breitendifferenz und dann um Längendifferenz drehen"];
xNew = goNew[xKS]; yNew = goNew[yKS]; zNew = goNew[zKS];
Print["xNew KS =", xNew // N];
Print["yNew KS =", yNew // N];
Print["zNew KS =", zNew // N];
solvU = Solve[uAeq == u1 xNew + u2 yNew + u3 zNew, {u1, u2, u3}] // Chop // Flatten //
  N; uRichtung = {u1, u2, u3} /. solvU;
Print["Kontrolle: gedrehtes u im neuen Parallelebenen-KS dargestellt ===> ",
 uRichtung, " 1. Koord. muss 0 sein"];
(* vAeq hat schon ins Ortsäquatorsyst. umgerechnete Koord.*)
solvV =
 Solve[vAeq == v1 xNew + v2 yNew + v3 zNew, {v1, v2, v3}] // Chop // Flatten // N;
vRichtung = {v1, v2, v3} /. solvV;
Print["Kontrolle: gedrehtes v im neuen Parallelebenen-KS dargestellt ===> ",
 vRichtung, " 1. Koord. muss 0 sein"];
vLen[vec_] := Sqrt[vec.vec]; (* ===> Vektorlänge *)
 (* ===> Vektorlänge *)
 (* ===> Drehung der Vertikalen *)
drehNewArc =
 -Sign[vRichtung[[2]]] ArcCos[vRichtung.zKS / (vLen[vRichtung] vLen[zKS])];
drehNewGrad = drehNewArc / (2 Pi) 360;
 vv = drehNewArc;
Print["drehNewGrad, ", drehNewGrad];
Print["vRichtung ", vRichtung];
Print["zKS ", zKS];
drehPlot = {{Cos[vv], -Sin[vv]}, {Sin[vv], Cos[vv]}};
Print["drehPlot = ", drehPlot // MatrixForm];
Print["Out-Variablen: Delta1, Delta1Grad,
   Delta2, Delta2Std, drehNewArc,drehNewGrad, drehPlot, vLen"];];
```

■ Beispiel (* INITIALISATION *)

```
Print[φ];
```

```
47.09
```

```
newPositionWithPrint[30, 6, φ]
```

```
u = {0.866025, 0.5, 0.}
```

```
v = {-0.0522642, 0.0905243, 0.994522}
```

$$w = \begin{pmatrix} 0.497261 \\ -0.861281 \\ 0.104528 \end{pmatrix}$$

```
d = 42.91
```

$$dreh = \begin{pmatrix} 1. & 0. & 0. \\ 0. & 0.732424 & -0.680849 \\ 0. & 0.680849 & 0.732424 \end{pmatrix}$$

$$r = \begin{pmatrix} 0.497261 \\ -0.701991 \\ -0.509843 \end{pmatrix}$$

```
r1 = {0.497261, -0.701991, -0.509843}

R = {0.701991, -0.497261, -0.509843}

Delta1Grad = -30.6534 Vorzeichen= -1

Delta2Grad = -35.3121, Delta2Std = -2.35414 Vorzeichen= -1

Test Matrixprodukt ===> True

Koordinatensystem : ( 1  0  0 )
                    ( 0  1  0 )
                    ( 0  0  1 )

Prozedur Koord.Umrechnung von - r1[[2]],-r1[[1]],r1[[3]] auf dreh.vec anwenden

===>vAeq = {0.610817, 0.0522642, 0.790045}

===>uAeq = {-0.366212, -0.866025, 0.340424}

===>xAeq = {0., -1., 0.}

===>yAeq = {-0.732424, 0., 0.680849}

===>zAeq = {0.680849, 0., 0.732424}

        ( 0.816015  0.578031  0. )
drehZ = (-0.578031  0.816015  0. )
        (    0.        0.     1. )

        ( 0.860267  0.  0.509843 )
drehY = (    0.     1.     0.    )
        (-0.509843  0.  0.860267 )

Vektor um Breitendifferenz und dann um Längendifferenz drehen

xNew KS ={0.701991, -0.497261, -0.509843}

yNew KS ={0.578031, 0.816015, 0.}

zNew KS ={0.41604, -0.294705, 0.860267}

Kontrolle: gedrehtes u im neuen Parallelebenen-KS dargestellt ===>
 {0., -0.918372, 0.395719} 1. Koord. muss 0 sein

Kontrolle: gedrehtes v im neuen Parallelebenen-KS dargestellt ===>
 {0., 0.395719, 0.918372} 1. Koord. muss 0 sein

drehNewGrad, -23.3108

vRichtung {0., 0.395719, 0.918372}

zKS {0, 0, 1}

drehPlot = ( 0.918372  0.395719 )
           (-0.395719  0.918372 )

Out-Variablen: Delta1, Delta1Grad, Delta2, Delta2Std, drehNewArc,drehNewGrad, drehPlot, vLen
```

■ Programm (* DEFINITION*)

```
newPosition[ostNord_, zenitNord_, φ_] :=
 Module[{a, b, φOld, u, v, w, d, r, r1, R, dY, dZ, xKS, yKS, zKS, goAeq, xNew, yNew,
   zNew, solvU, solvV, uRichtung, vRichtung, u1, u2, u3, v1, v2, v3, vv},
  (* Umrechnung mit Vektorprodukt *)
  (* ===> INPUT, Altgrad *)
  a = ostNord ; b = zenitNord ; φOld = φ;
  u = {Cos[a Degree], Sin[a Degree], 0};
  v = {-Sin[a Degree] Sin[b Degree], Cos[a Degree] Sin[b Degree], Cos[b Degree]};
  (* w: Normalenvektor im Ortshorizontsystem *)
  w = Transpose[{Cross[u, v]}];
  (* Drehung um Ost-West-Achse im Ortshorizontsystem *)
  (* Drehung ins Orstäquatorsystem *)
  d = (90 - φOld) ;
  dreh =
   {{1, 0, 0}, {0, Cos[d Degree], -Sin[d Degree]}, {0, Sin[d Degree], Cos[d Degree]}};
  r = dreh.w; r1 = Flatten[r];
  (* alte -y heisst x, alte -x heisst y *)
  R = {- r1[[2]], -r1[[1]], r1[[3]]};
  Delta1 = Sign[r1[[3]]]
    ArcCos[Sqrt[(R[[1]]^2 + R[[2]]^2) / (R[[1]]^2 + R[[2]]^2 + R[[3]]^2)]] ;
  (* Delta1 ist Winkel zwischen Vektor und Ortsäquatorebene,
   d.h. die neue geographische Breite *)
  Delta2 = Sign[r1[[3]]] ArcCos[Sqrt[R[[1]]^2 / (R[[1]]^2 + R[[2]]^2)]];
  (* Delta2 ist Winkel zwischen
    Vektorprojektion in Ortsäquatorebene x resp. Süd *)
  dY = Delta1; dZ = Delta2;
  Delta1Grad = Delta1 / Degree // N;
  Delta2Grad = Delta2 / (2 Pi) 360 // N;; Delta2Std = Delta2 / (2 Pi) 24 // N;
  Print["Delta1Grad = ", Delta1Grad, " Vorzeichen= " , Sign[r1[[3]]]];
  Print["Delta2Grad = ", Delta2Grad,
   ", Delta2Std = ", Delta2Std, " Vorzeichen= " , Sign[r1[[2]]]];
  (*Umrechnen des Vertikalvektors der Wand zur Drehung
    des Outputs in Vertikallage, in Einzelschritten*)
  xKS = {1, 0, 0}; yKS = {0, 1, 0}; zKS = {0, 0, 1};
  goAeq[vec_] := {-(dreh.vec)[[2]], -(dreh.vec)[[1]], (dreh.vec)[[3]]};
  vAeq = goAeq[v]; uAeq = goAeq[u];
  xAeq = goAeq[xKS]; yAeq = goAeq[yKS]; zAeq = goAeq[zKS];
  drehZ = {{Cos[dZ], -Sin[dZ], 0}, {Sin[dZ], Cos[dZ], 0}, {0, 0, 1}};
  drehY = {{Cos[dY], 0, -Sin[dY]}, {0, 1, 0}, {Sin[dY], 0, Cos[dY]}};
  goNew[vec_] := drehZ.(drehY.vec);
  xNew = goNew[xKS]; yNew = goNew[yKS]; zNew = goNew[zKS];
  solvU =
   Solve[uAeq == u1 xNew + u2 yNew + u3 zNew, {u1, u2, u3}] // Chop // Flatten // N;
  solvV = Solve[vAeq == v1 xNew + v2 yNew + v3 zNew, {v1, v2, v3}] // Chop // Flatten // N;
  uRichtung = {u1, u2, u3} /. solvU;
  vRichtung = {v1, v2, v3} /. solvV;
  vLen[vec_] := Sqrt[vec.vec];
  drehNewArc =
   -Sign[vRichtung[[2]]] ArcCos[vRichtung.zKS / (vLen[vRichtung] vLen[zKS])];
  drehNewGrad = drehNewArc / (2 Pi) 360;
  vv = drehNewArc;
```

```
Print["drehNewGrad = ", drehNewGrad];
drehPlot = {{Cos[vv], -Sin[vv]}, {Sin[vv], Cos[vv]}};
Print["drehPlot = ", drehPlot // MatrixForm];
Print["Out-Variablen: Delta1, Delta1Grad,
   Delta2, Delta2Std, drehNewArc,drehNewGrad, drehPlot, vLen"];]
```

■ Beispiele (* INITIALISATION *)

```
newPosition[30, 6, φ]
```

```
Delta1Grad = -30.6534 Vorzeichen= -1

Delta2Grad = -35.3121, Delta2Std = -2.35414 Vorzeichen= -1

drehNewGrad = -23.3108

drehPlot = ( 0.918372  0.395719
           -0.395719  0.918372)

Out-Variablen: Delta1, Delta1Grad, Delta2, Delta2Std, drehNewArc,drehNewGrad, drehPlot, vLen
```

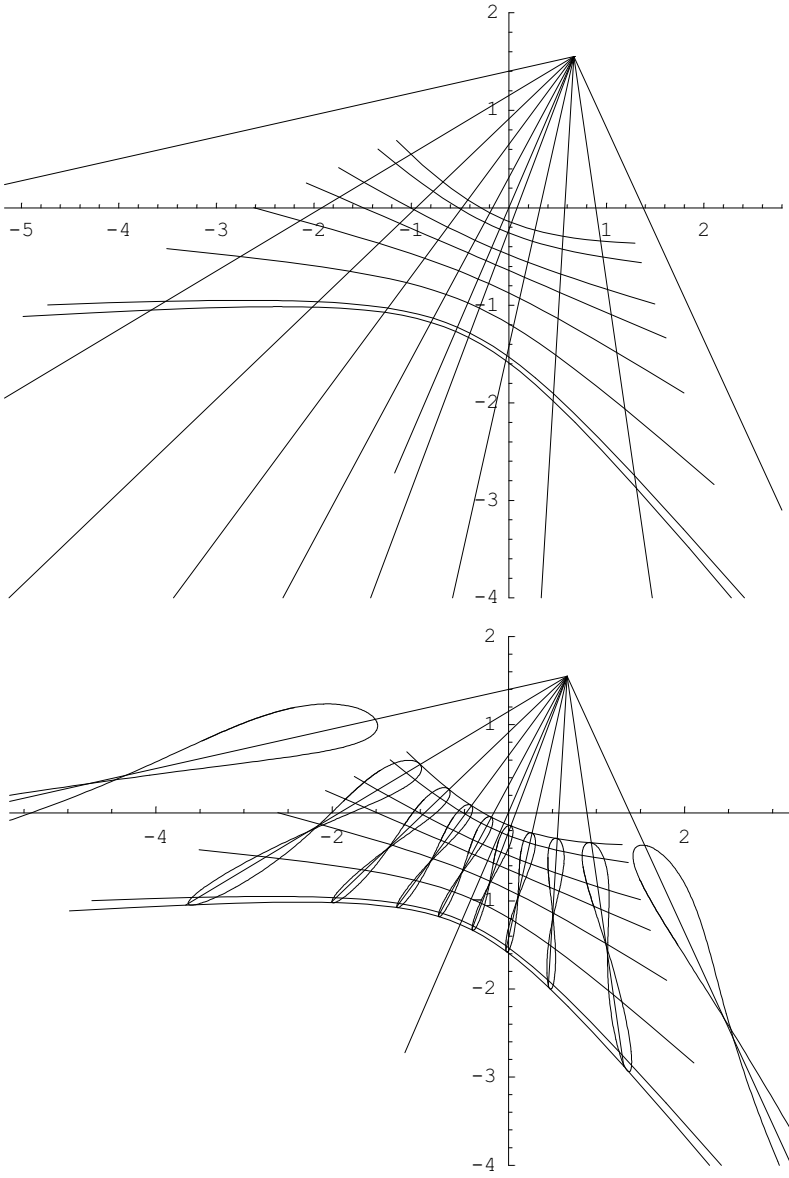
■ Plot-Programm

■ Programm

```
drehOmbrixDekSlPlot[{hMin_, hMax_}, {t1_, t2_}, {t3_, t4_}, {k1_, k2_}, {k3_, k4_},
   {pr1_, pr2_}, {faktor1_, faktor2_}, {nMitte_, nZiel_}, {nn1_, nn2_}] :=
  Module[{φNew, parPlotNow, k5, k6, tabd},
   φNew = Delta1Grad; fiNew = (90 - Abs[φNew]);
   k5 = Sign[k3]^2 k3 + (1 - Sign[k3]^2) hMin;
   k6 = Sign[k4]^2 k4 + (1 - Sign[k4]^2) hMax;
   v[gNow_, φNow_, dNow_, fiNow_, tNow_] :=
    gNow / (Cos[Sign[φNow] dNow Degree] Cos[tNow hour] Sin[fiNow Degree] +
        Sin[Sign[φNow] dNow Degree] Cos[fiNow Degree])
      {(Cos[Sign[φNow] dNow Degree] Sin[tNow hour]), Sign[φNow]
        ((Cos[Sign[φNow] dNow Degree] Cos[tNow hour] Cos[fiNow Degree] -
           Sin[Sign[φNow] dNow Degree] Sin[fiNow Degree]) )};
   vDreh[gNow_, φNow_, dNow_, fiNow_, tNow_] := drehPlot.
     v[gNow, φNow, dNow, fiNow, tNow];
   (*===> hMin,hMax eingeben!*)
   tabPlotPunkteDelta[φVar_, hMinInt_, hMaxInt_] :=
    Table[{MP[tVar - Delta2Std, φVar], g / Tan[(90 - φVar) Degree]},
     {tVar, hMinInt, hMaxInt, 1}];
   (* Drehombrix, Deklinationslinien*)
   ParametricPlot[Evaluate[Table[vDreh[g, φNew, d[k], fiNew, t], {k, k1, k2}]],
    {t, t1, t2}, AspectRatio → Automatic, PlotRange → {pr1, pr2} ,
    Epilog → Join[Table[Line[{drehPlot.ursprung[φNew], drehPlot.
          (faktor1 ( tabPlotPunkteDelta[φNew, hMin, hMax][[n]] - ursprung[φNew]) +
            ursprung[φNew])}],
       {n, 1, Length[tabPlotPunkteDelta[φNew, hMin, hMax]]}], {Line[{drehPlot.
          ursprung[φNew], faktor2 drehPlot.tabPlotPunkte[φNew][[nMitte]]}]}]];
   nn1 = Sign[nn1]^2 nn1 + (1 - Sign[nn1]^2) hMin;
   nn2 = Sign[nn2]^2 nn2 + (1 - Sign[nn2]^2) hMax;
   parPlotNow = ParametricPlot[
     Evaluate[Table[vDreh[g, φNew, d[k], fiNew, (t - Delta2Std)], {k, k1, k2}]], {t,
      t3 + Delta2Std, t4 + Delta2Std}, AspectRatio → Automatic, PlotRange → {pr1, pr2},
     Epilog -> {Join[Table[Line[{vDreh[g, φNew, d[nZiel], fiNew, k - Delta2Std],
           drehPlot.ursprung[φNew]}], {k, k5, k6, 1}],
       {Line[{drehPlot.ursprung[φNew], faktor2 drehPlot.
            tabPlotPunkte[φNew][[nMitte]]}]} ]}, DisplayFunction → Identity];
   tabd[n_] := Table[vDreh[g, φNew, ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)]
            Tan[eE[dd] / 2 ]] + L0 + L01 dd + L01 jahresTage[jahrOut]] Sin[Eps]] /
       Degree, fiNew, (n - zgStd[dd] - Delta2Std)], {dd, -100, 300}];
   listp[n_] := ListPlot[tabd[n], PlotJoined → True,
     AspectRatio → Automatic, DisplayFunction → Identity];
   listp0 = Show[Table[listp[nn], {nn, nn1, nn2}], DisplayFunction → Identity];
   Show[parPlotNow, listp0, DisplayFunction → $DisplayFunction];];
drehOmbrixDekSlPlot[hMin_, hMax_, t1_, t2_, t3_, t4_, k1_, k2_,
  k3_, k4_, pr1_, pr2_, faktor1_, faktor2_, nMitte_, nZiel_, nn1, nn2] :=
 drehOmbrixDekSlPlot[{hMin, hMax}, {t1, t2}, {t3, t4}, {k1, k2}, {k3, k4},
  {pr1, pr2}, {faktor1, faktor2}, {nMitte, nZiel}, {nn1, nn2},
  Print["Ende"];]
```

■ Beispiele

```
drehOmbrixDekSlPlot[{-7, 2}, {-4, 4},
 {-4, 4}, {1, 8}, {0, 0}, {-4, 2}, {4, 5}, {6, 7}, {-7, 2}]
```



Work

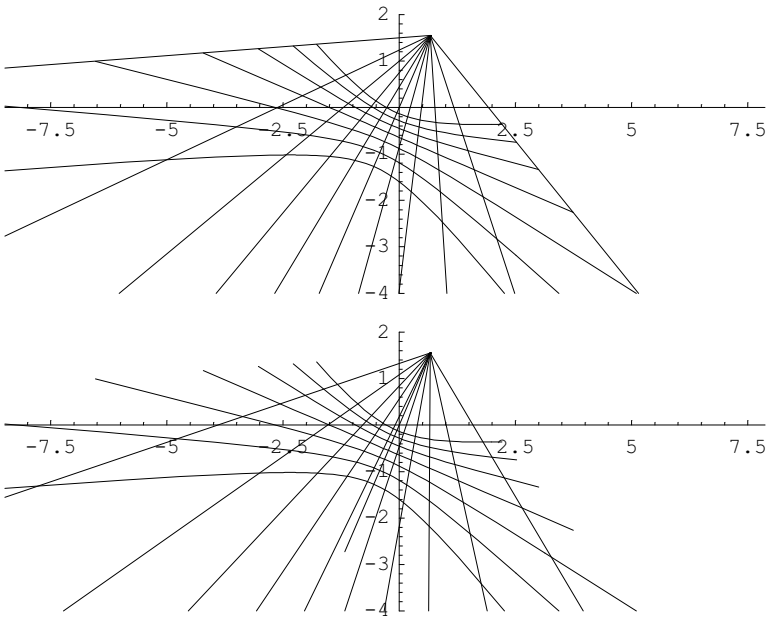
■ **Hier eine ältere Fassung:**

```
(* "Putzmaschine" *) (*Old Form:Remove["Global`@*"]*) (*  Remove["Global`*"]; *)
(* Umrechnung mit Vektorprodukt *)
(* ===> INPUT, Altgrad *)  a = 30 ; b = 6 ;  φ = 47.09; φ0 = φ;
dek[1] = 23.5 ; dek[2] = 16 ; dek[3] = 8 ; dek[4] = 0 ;
dek[5] = -8 ; dek[6] = -16 ; dek[7] = -23.5 ;  g = 1;

hour = 2 Pi / 24;
m0 = (356.5399 - 360) Degree; e = 0.0167092;
L0 = (282.9399 - 360) Degree; Eps = 23.4393 Degree;
L01 = 0.0000470684 Degree; m01 = 0.9856002664 Degree;
(* Keplerfunktion *)
f[eE_, d_] := eE - e Sin[eE] - m0 - m01 d;
eE[d_] := eNS /. FindRoot[f[eNS, d] == 0, {eNS, 3}];
(* Zeitgleichung *)zG[d_] := L0 + m0 + (L01 + m01) d -
   ArcTan[Tan[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 + L01 d] Cos[Eps]];
(* Die Sprünge des Arcustangens durch Stetigmachen
  resp. Zusammensetzen überwinden:*)
updown[d_] := (1 - Sign[(zG[d] - 1)]) / 2 zG[d];
upup[d_] := (1 + Sign[(zG[d] - 1)]) / 2 (zG[d] - Pi);
up[d_] := updown[d] + upup[d];
downall[d_] := (1 - Sign[(zG[d] + 1)]) / 2 (zG[d] + Pi);
upall[d_] := (1 + Sign[(zG[d] + 1)]) / 2 up[d];
glattZG[d_] := (upall[d] + downall[d]);
zgStd[d_] := glattZG[d] / (2 Pi) 24 ;

u = {Cos[a Degree], Sin[a Degree], 0};
v = {-Sin[a Degree] Sin[b Degree], Cos[a Degree] Sin[b Degree], Cos[b Degree]};
w = Transpose[{Cross[u, v]}];
d = (90 - φ) ;
dreh =
  {{1, 0, 0}, {0, Cos[d Degree], -Sin[d Degree]}, {0, Sin[d Degree], Cos[d Degree]}};
r = dreh.w; r1 = Flatten[r];
R = {- r1[[2]], -r1[[1]], r1[[3]]};
Delta1 = ArcCos[Sqrt[(R[[1]]^2 + R[[2]]^2) / (R[[1]]^2 + R[[2]]^2 + R[[3]]^2)]] ;
Delta2 = ArcCos[Sqrt[R[[1]]^2 / (R[[1]]^2 + R[[2]]^2)]]; dY = Delta1; dZ = Delta2;
Delta1Grad = Delta1 / Degree // N; Delta2Grad = Delta2 / (2 Pi) 360 // N;;
Delta2Std = Delta2 / (2 Pi) 24 // N;
Print["Delta1Grad = ", Delta1Grad,
  ", Delta2Grad = ", Delta2Grad, ", Delta2Std = ", Delta2Std];
(*Umrechnen des Vertikalvektors und zur Drehung des
  Outputs in Vertikallage in Einzelschritten*)
Print["Test Matrixprodukt ===> ", dreh.v == Flatten[dreh.Transpose[{v}]]];
goAeq[vec_] := {-(dreh.vec)[[2]], -(dreh.vec)[[1]], (dreh.vec)[[3]]};
xKS = {1, 0, 0}; yKS = {0, 1, 0}; zKS = {0, 0, 1};
```

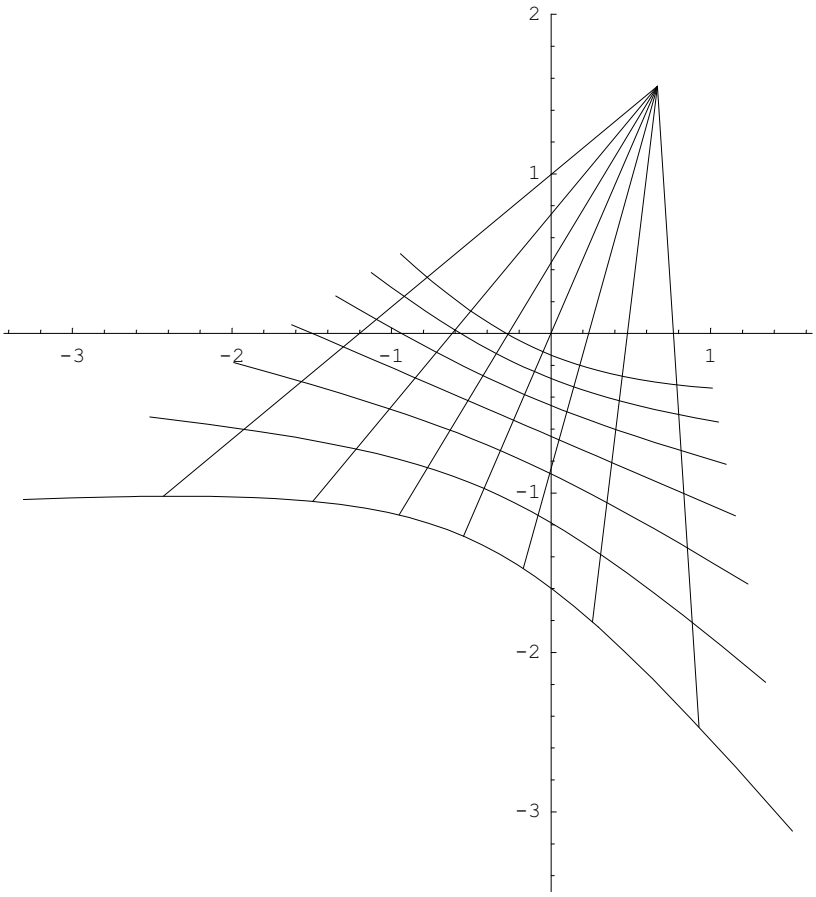
```
vAeq = goAeq[v]; uAeq = goAeq[u]; xAeq = goAeq[xKS];
yAeq = goAeq[yKS]; zAeq = goAeq[zKS];
drehZ = {{Cos[-dZ], -Sin[-dZ], 0}, {Sin[-dZ], Cos[-dZ], 0}, {0, 0, 1}};
drehY = {{Cos[-dY], 0, -Sin[-dY]}, {0, 1, 0}, {Sin[-dY], 0, Cos[-dY]}};
goNew[vec_] := drehZ.(drehY.vec);
xNew = goNew[xKS]; yNew = goNew[yKS]; zNew = goNew[zKS];
Print["Kontrolle 1. Koord 0 ===> ",
 solvU = Solve[uAeq == u1 xNew + u2 yNew + u3 zNew, {u1, u2, u3}] // Chop // Flatten]
Print["Kontrolle 1. Koord 0 ===> ",
  solvV = Solve[vAeq == v1 xNew + v2 yNew + v3 zNew, {v1, v2, v3}] // Chop // Flatten];
uRichtung = {u1, u2, u3} /. solvU;
vRichtung = {v1, v2, v3} /. solvV;
vLen[vec_] := Sqrt[vec.vec];
drehNewArc = ArcCos[vRichtung.zKS / (vLen[vRichtung] vLen[zKS])];
drehNewGrad = drehNewArc / (2 Pi) 360; Print["Drehung der Vertikalen ", drehNewGrad];
vv = -drehNewArc;
drehPlot = {{Cos[vv], -Sin[vv]}, {Sin[vv], Cos[vv]}};
Print["drehPlot = ", drehPlot // MatrixForm];
(* Plot *)

φ = - Delta1Grad; fi = (90 - Abs[φ]);
hour = 2 Pi / 24 (* Tage *);

MP[t_, φ_] := g Tan[t hour] / Sin[(90 - φ) Degree];
tab1[φ_] := Table[MP[t, φ], {t, -5, 5, 1}]; tab1[φ];
tab2[φ_] := Table[{tab1[φ][[n]], g / Tan[(90 - φ) Degree]}, {n, 1, Length[tab1[φ]]}];
tab2[φ];
vV[h_, fi_, t_] := drehPlot.
   (g / (Cos[Sign[φ] h Degree] Cos[t hour] Sin[fi Degree] + Sin[Sign[φ] h Degree]
         Cos[fi Degree]) {(Cos[Sign[φ] h Degree] Sin[t hour]),
      Sign[φ] ((Cos[Sign[φ] h Degree] Cos[t hour] Cos[fi Degree] -
          Sin[Sign[φ] h Degree] Sin[fi Degree]) )});

(* Ursprung versetzt, vgl. Gnomonstab parallel Erdachse! *)
ursprung[φ_] := {0, -g / Tan[φ Degree]};
ParametricPlot[Evaluate[Table[vV[dek[k], fi, t], {k, 1, 7}]],
  {t, -5, 5}, AspectRatio → Automatic, PlotRange → {-4, 2},
  Epilog -> {Table[Line[{drehPlot.ursprung[φ], drehPlot.
        (4 (tab2[φ][[n]] - ursprung[φ]) + ursprung[φ])}], {n, 1, Length[tab1[φ]]}]}];
tab3[φ_] := Table[MP[(t - Delta2Std), φ], {t, -2, 7, 1}]; tab3[φ];
tab4[φ_] := Table[{tab3[φ][[n]], g / Tan[(90 - φ) Degree]}, {n, 1, Length[tab3[φ]]}];
tab3[φ];
ParametricPlot[Evaluate[Table[vV[dek[k], fi, t], {k, 1, 7}]],
  {t, -5, 5}, AspectRatio → Automatic, PlotRange → {-4, 2},
  Epilog → Join[Table[Line[{drehPlot.ursprung[φ], drehPlot.
        (4 (tab4[φ][[n]] - ursprung[φ]) + ursprung[φ])}], {n, 1, Length[tab3[φ]]}],
    {Line[{drehPlot.ursprung[φ], 5 drehPlot.tab2[φ][[6]]}]}]];
Print["tab1[φ]:= ", tab1[φ], " Plot von -5 bis 5 Std. versetzte Zeit "];
Print["tab3[φ]:= ", tab3[φ],
 " Plot von -3-Delta2Std bis 7-Delta2Std korrigierte Zeit."]

(* Plot ohne korrigierte Zeit *)
parPlot = ParametricPlot[Evaluate[Table[vV[dek[k], fi, t], {k, 1, 7}]],
   {t, -3.5, 3.5}, AspectRatio → Automatic, PlotRange → {-3.5, 2},
   Epilog -> {Table[Line[{vV[dek[1], fi, k], drehPlot.ursprung[φ]}], {k, -3, 3, 1}]}];
```

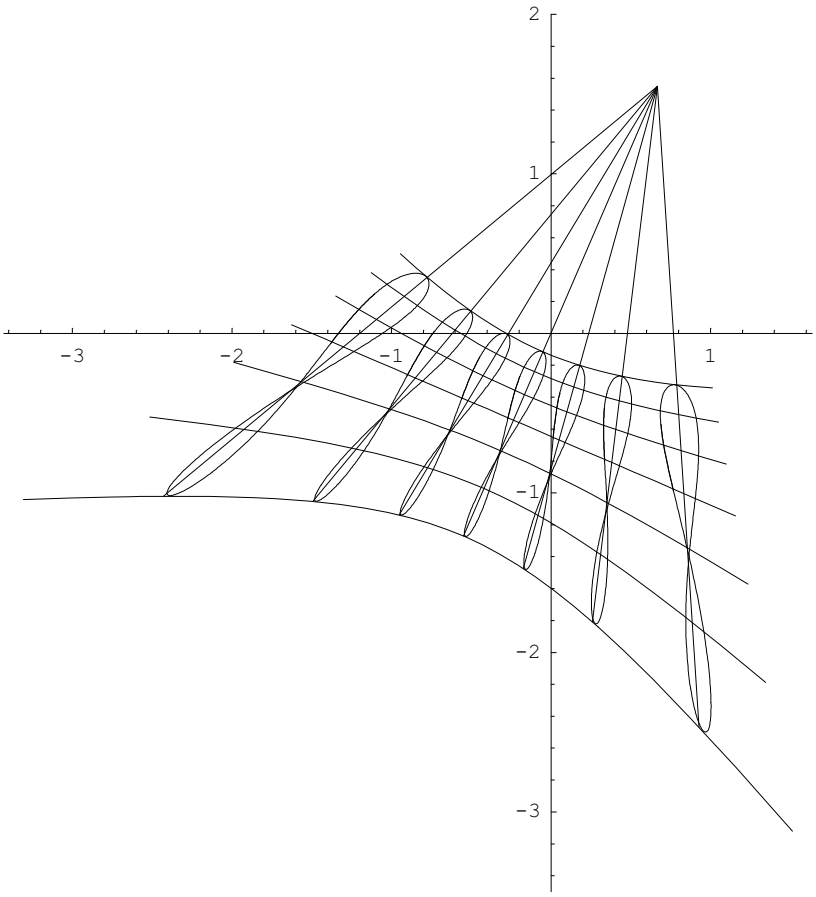
```
tabd[n_] := Table[
   vV[ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 + L01 d] Sin[Eps]] /
     Degree, fi, (n - zgStd[d])], {d, -100, 300}];
listp[n_] := ListPlot[tabd[n], PlotJoined → True, AspectRatio → Automatic,
   DisplayFunction → Identity];
listp0 = Show[listp[3], listp[2], listp[1], listp[0], listp[-1],
   listp[-2], listp[-3], DisplayFunction → Identity];
Show[parPlot, listp0, DisplayFunction → $DisplayFunction];
Print["Das waren Plots ohne korrigierte Zeit "];
Print["  "];
Print["Nachfolgend kommen Plots mit korrigierter Zeit "];

(* Plot mit korrigierter Zeit *)
parPlot = ParametricPlot[Evaluate[Table[vV[dek[k], fi, (t - Delta2Std)], {k, 1, 7}]],
   {t, -3.5 + Delta2Std, 3.5 + Delta2Std},
   AspectRatio → Automatic, PlotRange → {-3.5, 2},
   Epilog -> {Table[Line[{vV[dek[1], fi, k - Delta2Std], drehPlot.ursprung[φ]}],
      {k, -1, 5, 1}]}];
tabd[n_] := Table[vV[ArcSin[Sin[2 ArcTan[Sqrt[(1 + e) / (1 - e)] Tan[eE[d] / 2 ]] + L0 +
        L01 d] Sin[Eps]] / Degree, fi, (n - zgStd[d] - Delta2Std)], {d, -100, 300}];
listp[n_] := ListPlot[tabd[n], PlotJoined → True, AspectRatio → Automatic,
   DisplayFunction → Identity];
listp0 = Show[listp[5], listp[4], listp[3], listp[2], listp[1],
   listp[0], listp[-1], DisplayFunction → Identity];
Show[parPlot, listp0, DisplayFunction → $DisplayFunction];
Print["Das waren Plots mit korrigierter Zeit "];
```

```
Delta1Grad = 30.6534, Delta2Grad = 35.3121, Delta2Std = 2.35414

Test Matrixprodukt ===> True

Kontrolle 1. Koord 0 ===> {u1 → 0, u2 → -0.918372, u3 → 0.395719}

Kontrolle 1. Koord 0 ===> {v1 → 0, v2 → 0.395719, v3 → 0.918372}

Drehung der Vertikalen 23.3108

drehPlot = ( 0.918372  0.395719
            -0.395719  0.918372 )
```



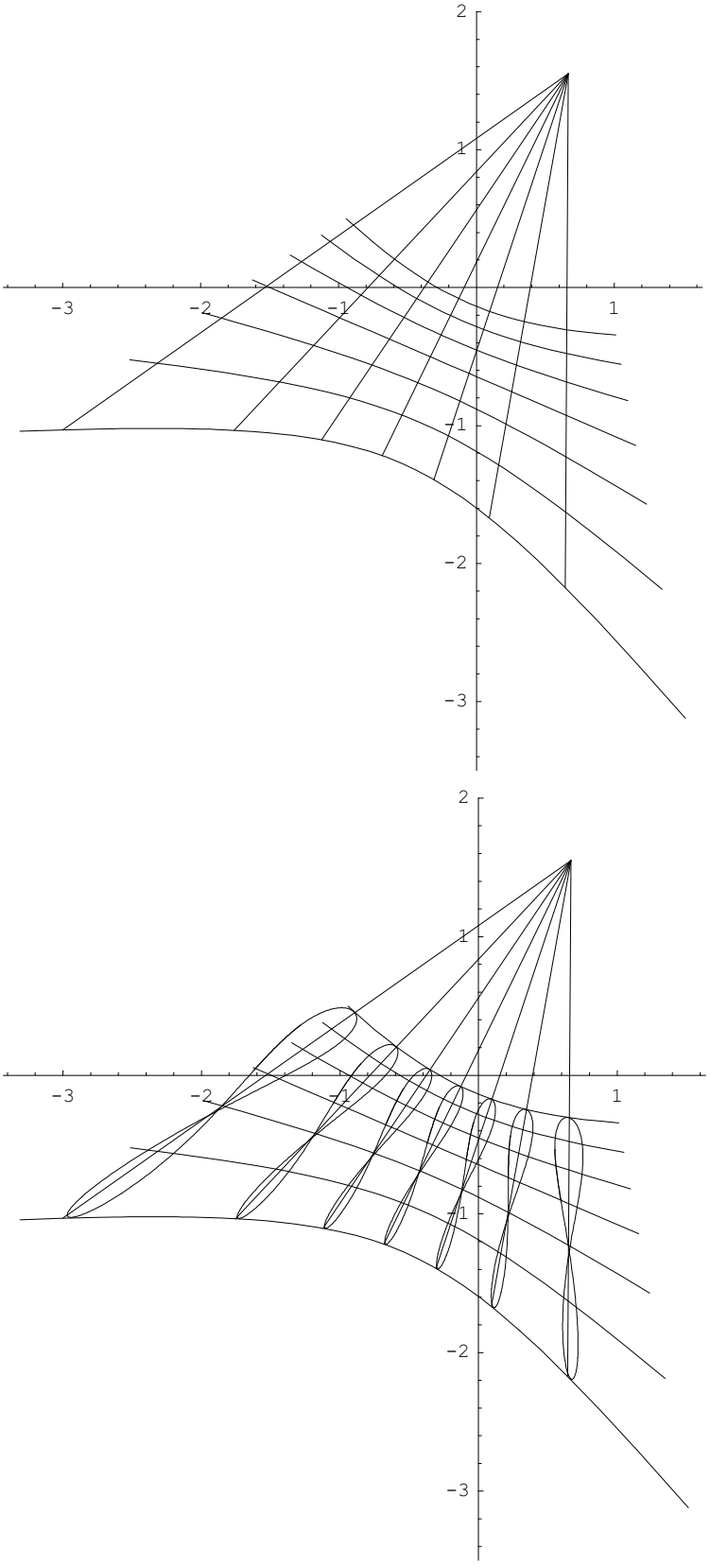
```
tab1[φ]:= {-4.33824, -2.01339, -1.16243, -0.671129, -0.311472, 0, 0.311472,
  0.671129, 1.16243, 2.01339, 4.33824} Plot von -5 bis 5 Std. versetzte Zeit

tab3[φ]:= {-2.52871, -1.40076, -0.823416, -0.430276, -0.108084, 0.198444, 0.534359,
  0.964651, 1.64102, 3.14041} Plot von -3-Delta2Std bis 7-Delta2Std korrigierte Zeit.
```





Das waren Plots ohne korrigierte Zeit

Nachfolgend kommen Plots mit korrigierter Zeit



Das waren Plots mit korrigierter Zeit