The `tikz-3dplot-circleofsphere` Package: Drawing circles of a sphere with `tikz-3dplot`

Matthias Wolff [0000–0002–3895–7313]

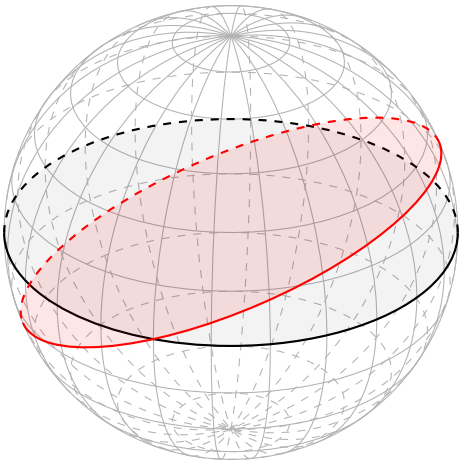
BTU Cottbus-Senftenberg

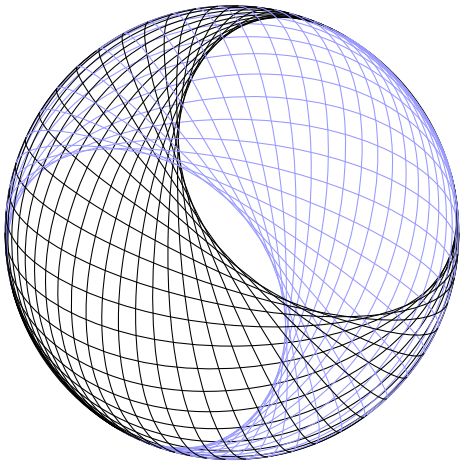
August 22, 2018

See https://github.com/matthias-wolff/tikz-3dplot-circleofsphere/blob/master/tikz-3dplot-circleofsphere.pdf for the latest version of this document.

Abstract

A *circle of a sphere* is a circle drawn on a spherical surface like, for instance, circles of latitude or longitude. Circles in arbitrary 3D positions can be drawn with TikZ [2] very easily using a transformed coordinate system provided by the `tikz-3dplot` package [1] (that is because TikZ can only draw circles on the xy-plane). However, automatically distinguishing the parts of the circle lying on the front and back sides of the sphere, e.g. by drawing a solid arc on the front side and a dashed one on the back side, is a somewhat tricky feat. The `tikz-3dplot-circleofsphere` package will perform that feat for you.





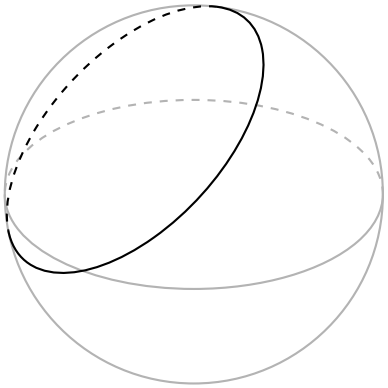
```
\documentclass{standalone}
\usepackage{tikz-3dplot-circleofsphere}
\begin{document}
  \centering
  \def\r{3}
  \tdplotsetmaincoords{60}{125}
  \begin{tikzpicture}[tdplot_main_coords]
    \draw[tdplot_screen_coords,thin,black!30] (0,0,0) circle (\r);
    \foreach \a in {-75,-60,...,75}
      {\tdplotCsDrawLatCircle[thin,black!30]{\r}{\a}}
    \foreach \a in {0,15,...,165}
      {\tdplotCsDrawLonCircle[thin,black!30]{\r}{\a}}
    \tdplotCsDrawLatCircle%
      [thick,tdplotCsFill/.style={opacity=0.05}]{\r}{0}
    \tdplotCsDrawGreatCircle%
      [red,thick,tdplotCsFill/.style={opacity=0.1}]{\r}{105}{-23.5}
  \end{tikzpicture}
\end{document}
```

```
\documentclass{standalone}
\usepackage{tikz-3dplot-circleofsphere}
\begin{document}
  \centering
  \def\r{3}
  \tdplotsetmaincoords{60}{125}
  \begin{tikzpicture}[tdplot_main_coords]
    \def\e{80};
    \draw[tdplot_screen_coords,very thin] (0,0,0) circle (\r);
    \foreach \a in {0,5,...,175} {
      \tdplotCsDrawGreatCircle%
        [thin, tdplotCsBack/.style={thin,blue!40}]%
        {\r}{\a}{90*sin(\a)*sin(\e)}
    }
  \end{tikzpicture}
\end{document}
```

Contents

1 Just Looking for the Minimalist Code?

There you go!



```
\documentclass{standalone}
\usepackage{tikz,tikz-3dplot}
%% >> MINIMALIST CIRCLE OF SHPERE DRAWING CODE ____________________________________________________________________________
\newcommand\scircle[4]{%
  \tdplotsetrotatedcoords{#2}{#3}{0}                                    % Rotate coordinate system
  \let\a\tdplotalpha                                                    % alpha (rotated coord. system)
  \let\b\tdplotbeta                                                     % beta (rotated coord. system)
  \let\p\tdplotmainphi                                                  % phi (main coord. system)
  \let\t\tdplotmaintheta                                                % theta (main coord. system)
  \pgfmathsetmacro\azx{cos(\a)*cos(\b)*sin(\p)*sin(\t) - sin(\b)*cos(\t) - cos(\b)*cos(\p)*sin(\a)*sin(\t)}
  \pgfmathsetmacro\azy{-cos(\a)*cos(\p)*sin(\t) - sin(\a)*sin(\p)*sin(\t)}
  \pgfmathsetmacro\azz{cos(\b)*cos(\t) + cos(\a)*sin(\b)*sin(\p)*sin(\t) - cos(\p)*sin(\a)*sin(\b)*sin(\t)}
  \pgfmathsetmacro\re {#1*cos(#4)}                                      % Radius of circle
  \pgfmathsetmacro\ze {#1*sin(#4)}                                      % z-coordinate of drawing plane
  \pgfmathsetmacro\coX{\ze*cos(#2)*sin(#3)}                             % x-coordinate offset for ze
  \pgfmathsetmacro\coY{\ze*sin(#2)*sin(#3)}                             % y-coordinate offset for ze
  \pgfmathsetmacro\coZ{\ze*cos(#3)}                                     % z-coordinate offset for ze
  \coordinate (coffs) at (\coX,\coY,\coZ);                              % Offset as coordinate value
  \tdplotsetrotatedcoordsorigin{(coffs)}                                % Offset coordinate system
  \begin{scope}[tdplot_rotated_coords]                                  % Drawing scope >>
    \pgfmathsetmacro\tanEps{tan(#4)}                                    %   Tangent of elevation angle
    \pgfmathsetmacro\bOneside{((\tanEps)^2)>=(((\azx)^2+(\azy)^2)/(\azz)^2)}    %   Circle entirely on one side?
    \ifthenelse{\bOneside=1}{%                                          %   Circle on one side of sphere >>
      \pgfmathsetmacro\bFrontside{(\azx*\re+\azz*\ze)>=0}                %     Circle entirely on front side?
       \ifthenelse{\bFrontside=1}                                       %     |
         {\draw (0,0) circle (\re);}                                    %     Draw on front side
         {\draw[dashed] (0,0) circle (\re);}                            %     Draw on back side
    }{%                                                                 %   << Circle on both sides >>
      \pgfmathsetmacro\u{\azy}                                          %     Substitution u=...
      \pgfmathsetmacro\v{sqrt( (\azx)^2 + (\azy)^2 - (\azz)^2*(\tanEps)^2 )}    %     Substitution v=...
      \pgfmathsetmacro\w{\azx - \azz*\tanEps}                           %     Substitution w=...
      \pgfmathsetmacro\phiBf{2*atan2(\u-\v,\w)}                         %     Back->front crossing angle
      \pgfmathsetmacro\phiFb{2*atan2(\u+\v,\w)}                         %     Front->back crossing angle
      \pgfmathsetmacro\bUnwrapA{(\phiFb-\phiBf)>360}                    %     Unwrap front->back angle #1?
      \pgfmathsetmacro\bUnwrapB{\phiBf>\phiFb}                          %     Unwrap front->back angle #2?
      \ifthenelse{\bUnwrapA=1}{\pgfmathsetmacro\phiBf{\phiBf+360}}{}    %     Unwrap front->back angle #1
      \ifthenelse{\bUnwrapB=1}{\pgfmathsetmacro\phiBf{\phiBf-360}}{}    %     Unwrap front->back angle #2
      \draw[dashed] (\phiFb:\re) arc (\phiFb:{\phiBf+360}:\re);         %     Draw back side arc
      \draw (\phiBf:\re) arc (\phiBf:\phiFb:\re);                       %     Draw back side arc
    }                                                                   %   <<
  \end{scope}                                                           % << (Drawing scope)
}
%% << ____________________________________________________________________________________________________________________
\begin{document}
  \tdplotsetmaincoords{60}{125}                                         % Set main coordintate system
  \begin{tikzpicture}[thick,tdplot_main_coords]                         % TikZ picture >>
    \begin{scope}[black!30]                                             %   Draw in gray >>
      \draw[tdplot_screen_coords] (0,0,0) circle (2.5);                 %    Sphere outline
      \scircle{2.5}{0}{0}{0}                                            %    Equator
    \end{scope}                                                         %   <<
    \scircle{2.5}{-40}{40}{30}                                          %   Draw another sphere circle
  \end{tikzpicture}                                                     % <<
\end{document}
```

Want some more convenience or interested in what we did? Read on...

2 The tikz-3dplot-circleofsphere Package

2.1 Installation

Download `tikz-3dplot-circleofsphere.sty` from [3] file into your project folder and include the package with `\usepackage{tikz-3dplot-circleofsphere}`.

2.2 Drawing Commands

`\tdplotCsDrawCircle[style]{r}{alpha}{beta}{epsilon}`

Draws a circle of a sphere.

Parameters

`style` TikZ style

- use `tdplotCsFront/.style={...}` to style the front side arc
- use `tdplotCsBack/.style={...}` to style the back side arc
- use `tdplotCsFill/.style={...}` to style the circle filling
- use `tdplotCsDrawAux` to draw some auxiliary information

`r` Radius of sphere

`alpha` Azimuthal angle of drawing plane.
Passed as `alpha` to `\tdplotsetrotatedcoords{alpha}{beta}{gamma}`

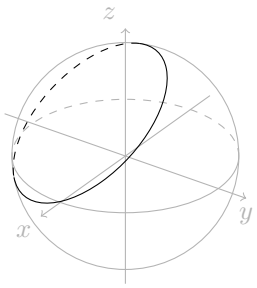
`beta` Polar angle of drawing plane.
Passed as `beta` to `\tdplotsetrotatedcoords{alpha}{beta}{gamma}`

`epsilon` Elevation angle of circle above the drawing plane. Permissible values are $-90 <$ `epsilon` < 90. Use 0 for drawing a great circle.

Output

–none–

Example




```
\def\r{1.5}
\tdplotsetmaincoords{60}{125}
\begin{tikzpicture}[tdplot_main_coords]
  \begin{scope}[thin,black!30]
    \draw[->] (-1.3*\r,0,0) -- (1.3*\r,0,0) node[anchor=north east] {$x$};
    \draw[->] (0,-1.3*\r,0) -- (0,1.3*\r,0) node[anchor=north] {$y$};
    \draw[->] (0,0,-1.3*\r) -- (0,0,1.3*\r) node[anchor=south east] {$z$};
    \draw[tdplot_screen_coords] (0,0,0) circle (\r);
    \tdplotCsDrawLatCircle{\r}{0}
  \end{scope}
  \tdplotCsDrawCircle{\r}{-40}{40}{30}
\end{tikzpicture}
```

\tdplotCsDrawGreatCircle[style]{r}{alpha}{beta}

Draws a great circle.
Equivalent to \tdplotCsDrawCircle[style]{r}{alpha}{beta}{0}.

Parameters

`style` TikZ style

- use `tdplotCsFront/.style={...}` to style the front side arc
- use `tdplotCsBack/.style={...}` to style the back side arc
- use `tdplotCsFill/.style={...}` to style the circle filling
- use `tdplotCsDrawAux` to draw some auxiliary information

`r` Radius of sphere

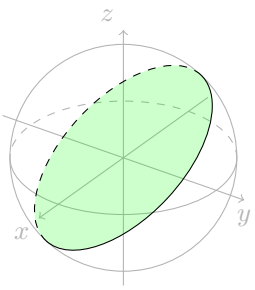
`alpha` Azimuthal angle of drawing plane.
Passed as `alpha` to \tdplotsetrotatedcoords{alpha}{beta}{gamma}

`beta` Polar angle of drawing plane.
Passed as `beta` to \tdplotsetrotatedcoords{alpha}{beta}{gamma}

Output

–none–

Example




```
\def\r{1.5}
\tdplotsetmaincoords{60}{125}
\begin{tikzpicture}[tdplot_main_coords]
  \begin{scope}[thin,black!30]
    \draw[->] (-1.3*\r,0,0) -- (1.3*\r,0,0) node[anchor=north east] {$x$};
    \draw[->] (0,-1.3*\r,0) -- (0,1.3*\r,0) node[anchor=north] {$y$};
    \draw[->] (0,0,-1.3*\r) -- (0,0,1.3*\r) node[anchor=south east] {$z$};
    \draw[tdplot_screen_coords] (0,0,0) circle (\r);
    \tdplotCsDrawLatCircle{\r}{0}
  \end{scope}
  \tdplotCsDrawGreatCircle[tdplotCsFill/.style={green,opacity=0.2}]{\r}{-40}{40}
\end{tikzpicture}
```

\tdplotCsDrawLatCircle[style]{r}{epsilon}

Draws a circle of latitude.
Equivalent to \tdplotCsDrawCircle[style]{r}{0}{0}{epsilon}.

Parameters

`style` TikZ style

- use `tdplotCsFront/.style={...}` to style the front side arc
- use `tdplotCsBack/.style={...}` to style the back side arc
- use `tdplotCsFill/.style={...}` to style the circle filling
- use `tdplotCsDrawAux` to draw some auxiliary information

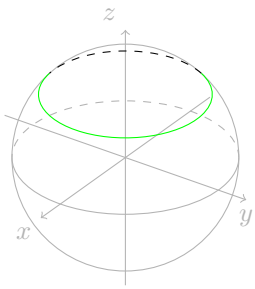
`r` Radius of sphere

`epsilon` Elevation angle of circle above the drawing plane. Permissible values are $-90 <$ `epsilon` < 90. Use 0 for drawing the sphere equator.

Output

–none–

Example




```
\def\r{1.5}
\tdplotsetmaincoords{60}{125}
\begin{tikzpicture}[tdplot_main_coords]
  \begin{scope}[thin,black!30]
    \draw[->] (-1.3*\r,0,0) -- (1.3*\r,0,0) node[anchor=north east] {$x$};
    \draw[->] (0,-1.3*\r,0) -- (0,1.3*\r,0) node[anchor=north] {$y$};
    \draw[->] (0,0,-1.3*\r) -- (0,0,1.3*\r) node[anchor=south east] {$z$};
    \draw[tdplot_screen_coords] (0,0,0) circle (\r);
    \tdplotCsDrawLatCircle{\r}{0}
  \end{scope}
  \tdplotCsDrawLatCircle[tdplotCsFront/.style={green}]{\r}{40}
\end{tikzpicture}
```

`\tdplotCsDrawLonCircle[style]{r}{alpha}`

Draws a circle of longitude.
Equivalent to `\tdplotCsDrawCircle[style]{r}{alpha}{90}{0}`.

Parameters

`style` TikZ style

- use `tdplotCsFront/.style={...}` to style the front side arc
- use `tdplotCsBack/.style={...}` to style the back side arc
- use `tdplotCsFill/.style={...}` to style the circle filling
- use `tdplotCsDrawAux` to draw some auxiliary information

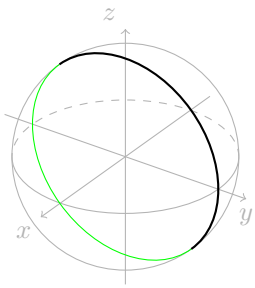
`r` Radius of sphere

`alpha` Azimuthal angle of drawing plane.
Passed as `alpha` to `\tdplotsetrotatedcoords{alpha}{beta}{gamma}`

Output

–none–

Example




```
\def\r{1.5}
\tdplotsetmaincoords{60}{125}
\begin{tikzpicture}[tdplot_main_coords]
  \begin{scope}[thin,black!30]
    \draw[->] (-1.3*\r,0,0) -- (1.3*\r,0,0) node[anchor=north east] {$x$};
    \draw[->] (0,-1.3*\r,0) -- (0,1.3*\r,0) node[anchor=north] {$y$};
    \draw[->] (0,0,-1.3*\r) -- (0,0,1.3*\r) node[anchor=south east] {$z$};
    \draw[tdplot_screen_coords] (0,0,0) circle (\r);
    \tdplotCsDrawLatCircle{\r}{0}
  \end{scope}
  \tdplotCsDrawLonCircle[thick,tdplotCsBack/.style={thin,solid,green}]{\r}{0}
\end{tikzpicture}
```

`\tdplotCsDrawPoint[style]{r}{alpha}{beta}`

Draws a point on a sphere.

Parameters

`style`	TikZ style • use `tdplotPtFront/.style={...}` to style a front side point • use `tdplotPtBack/.style={...}` to style a back side point
`r`	Radius of sphere
`alpha`	Azimuthal angle of drawing plane. Passed as `alpha` to `\tdplotsetrotatedcoords{alpha}{beta}{gamma}`
`beta`	Polar angle of drawing plane. Passed as `beta` to `\tdplotsetrotatedcoords{alpha}{beta}{gamma}`

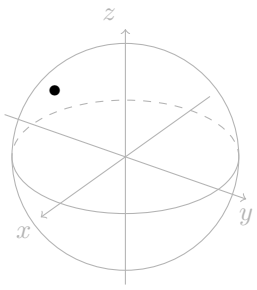
Output

–none–

Remarks

- Redefine `\tdplotCsFrontsidePoint` to customize drawing of a front side point.
- Redefine `\tdplotCsBacksidePoint` to customize drawing of a back side point.

Example




```
\def\r{1.5}
\tdplotsetmaincoords{60}{125}
\begin{tikzpicture}[tdplot_main_coords]
  \begin{scope}[thin,black!30]
    \draw[->] (-1.3*\r,0,0) -- (1.3*\r,0,0) node[anchor=north east] {$x$};
    \draw[->] (0,-1.3*\r,0) -- (0,1.3*\r,0) node[anchor=north] {$y$};
    \draw[->] (0,0,-1.3*\r) -- (0,0,1.3*\r) node[anchor=south east] {$z$};
    \draw[tdplot_screen_coords] (0,0,0) circle (\r);
    \tdplotCsDrawLatCircle{\r}{0}
  \end{scope}
  \tdplotCsDrawPoint{\r}{-40}{40}
\end{tikzpicture}
```

2.3 Auxiliary Commands

`\tdplotCsFrontsidePoint`

Invoked by `\tdplotCsDrawPoint` to draw a point on the front side of a sphere. Redefine to customize.

`\tdplotCsBacksidePoint`

Invoked by `\tdplotCsDrawPoint` to draw a point on the back side of a sphere. Redefine to customize.

`\tdplotCsComputeTransformRotScreen`

Computes the elements of the full rotation matrix

$$A = \begin{pmatrix} a_{xx} & a_{xy} & a_{xz} \\ a_{yx} & a_{yy} & a_{yz} \\ a_{zx} & a_{zy} & a_{zz} \end{pmatrix}.$$

See Section 3.1 for details.

Parameters

none

Output

`\axx`	Element a_{xx} of full rotation matrix
`\axy`	Element a_{xy} of full rotation matrix
...	
`\azz`	Element a_{zz} of full rotation matrix

Remarks

The command uses some internal variables of `tikz-3dplot`, namely `\tdplotalpha`, `\tdplotbeta`, `\tdplotmainphi`, and `\tdplotmaintheta`.

2.4 Known Issues

- The `tdplotCsFill` and `tdplotCsDrawAux` styles are only effective when specified directly with the drawing command.

3 Implementation Details

3.1 The Maths

Circles on a Sphere

We consider circles on a sphere of radius r as illustrated in Fig. 1. For drawing a great circle, i.e. a circle whose center coincides with the center of the sphere (blue in Fig. 1), we rotate the coordinate system by two Euler angles, an azimuthal angle $0° \leq \alpha < 360°$ and a polar angle $0° \leq \beta < 360°$, and draw on the new $x_r y_r$ plane. For drawing small circles (red in Fig. 1), we additionally elevate the drawing plane by an angle $-90° < \epsilon < 90°$, $\epsilon \neq 0$, and draw on the rotated and elevated $x_{ro} y_{ro}$ plane. The tricky part of drawing circles of spheres is to determine which part is on the back side of the sphere and to draw it a different style, e.g., dashed.

Coordinate Transforms with `tikz-3dplot`

We use the `circle` and `arc` path construction operations of TikZ for drawing. As TikZ will only draw circles and arcs on the xy-plane, we need to rotate and possibly offset the coordinate system as described above using the `tikz-3dplot` [1] package.

First, `tikz-3dplot` provides a *main coordinate system* which is basicly defining the view point on a 3D coordinate system. Denote by $P = (x\, y\, z)^\top$ a point in the 3D coordinate system. `tikz-3dplot` transforms that point in to screen coordinates $P' = (x'\, y'\, z')^\top$ by

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R^d(\phi, \theta) \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{1}$$

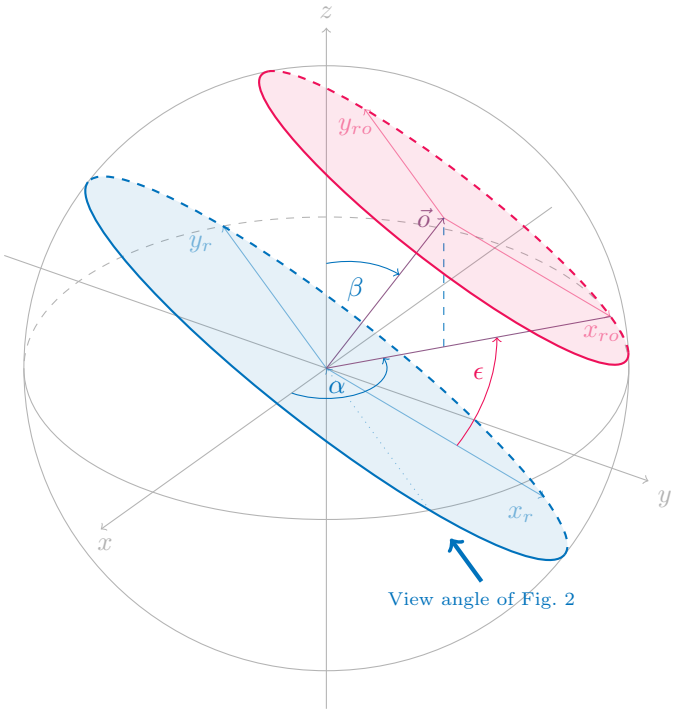


Figure 1: A great circle (blue) and a small circle (red).

with the rotation matrix[1]

$$
\begin{aligned}
R^d(\phi,\theta) &= \left(R^{z'}(\phi)\, R^x(\theta)\right)^\top \\
&= \left(\begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}\right)^\top \\
&= \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\cos\theta\sin\phi & \cos\theta\cos\phi & +\sin\theta \\ \sin\theta\sin\phi & -\sin\theta\cos\phi & \cos\theta \end{pmatrix}.
\end{aligned} \tag{2}
$$

We set the main coordinate system by `\tdplotsetmaincoords{⟨ϕ⟩}{⟨θ⟩}`.

Second, for drawing circles and arcs outside the xy-plane, we need to rotate the coordinate system further. To this end, we use `tikz-3dplot`'s *rotated coordinate system*[2]

$$
\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R^d(\phi,\theta)\, D(\alpha,\beta,\gamma) \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3}
$$

with the rotation matrix (cf. [1, p. 7])

$$
\begin{aligned}
D(\alpha,\beta,0) &= R^z(\alpha) R^y(\beta) \\
&= \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \\
&= \begin{pmatrix} \cos\alpha\cos\beta & -\sin\alpha & \cos\alpha\sin\beta \\ \sin\alpha\cos\beta & \cos\alpha & \sin\alpha\sin\beta \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}
\end{aligned} \tag{4}
$$

[1] Eqn. (2.1) in [1] seems to be incorrect. I used a version with changes marked in red: Since $\left(R^{z'}(\phi)\, R^x(\theta)\right)^\top = R^x(\theta)^\top\, R^{z'}(\phi)^\top$, rotations are performed in reverse order and direction.

[2] Eqn. (2.4) in [1] seems to be incorrect. I used a version with changes marked in red: Rotations are performed in reverse order.

where we deliberately omitted the last rotation $R^z(\gamma)$ by choosing $\gamma = 0$. Thus, the full rotation matrix for drawing a great circle is

$$
\begin{aligned}
A = \begin{pmatrix} a_{xx} & a_{xy} & a_{xz} \\ a_{yx} & a_{yy} & a_{yz} \\ a_{zx} & a_{zy} & a_{zz} \end{pmatrix} &= R^d(\phi,\theta)\, D(\alpha,\beta,0) \qquad (5)\\
&= \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\cos\theta\sin\phi & \cos\theta\cos\phi & \sin\theta \\ \sin\theta\sin\phi & -\sin\theta\cos\phi & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\alpha\cos\beta & -\sin\alpha & \cos\alpha\sin\beta \\ \sin\alpha\cos\beta & \cos\alpha & \sin\alpha\sin\beta \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \\
&= \begin{pmatrix} \cos\alpha\cos\beta\cos\phi + \cos\beta\sin\alpha\sin\phi \\ \cos\beta\cos\phi\sin\alpha\cos\theta - \cos\alpha\cos\beta\cos\theta\sin\phi - \sin\beta\sin\theta \\ \cos\alpha\cos\beta\sin\phi\sin\theta - \sin\beta\cos\theta - \cos\beta\cos\phi\sin\alpha\sin\theta \end{pmatrix} \\
&\qquad \begin{matrix} \cos\alpha\sin\phi - \cos\phi\sin\alpha \\ \cos\alpha\cos\phi\cos\theta + \sin\alpha\cos\theta\sin\phi \\ -\cos\alpha\cos\phi\sin\theta - \sin\alpha\sin\phi\sin\theta \end{matrix} \\
&\qquad \begin{matrix} \cos\alpha\cos\phi\sin\beta + \sin\alpha\sin\beta\sin\phi \\ \cos\beta\sin\theta - \cos\alpha\sin\beta\cos\theta\sin\phi + \cos\phi\sin\alpha\sin\beta\cos\theta \\ \cos\beta\cos\theta + \cos\alpha\sin\beta\sin\phi\sin\theta - \cos\phi\sin\alpha\sin\beta\sin\theta \end{matrix} \end{pmatrix}
\end{aligned}
$$

The rotated coordinate system is set by `\tdplotsetrotatedcoords{`$\langle\alpha\rangle$`}{`$\langle\beta\rangle$`}{0}`.

With the coordinate transforms described so far, we can only draw great circles. For drawing small circles, we additionally need to offset the origin of the rotated coordinate system. To this end we define an elevation angle ϵ which defines the height

$$z_e = r\, \sin\epsilon \qquad (6)$$

of the offset drawing plane over the rotated one as illustrated in Fig. 2 (r is still the radius of the sphere). As the circle to be drawn must lie on the sphere, its radius

$$r_e = r\, \cos\epsilon \qquad (7)$$

decreases with increasing elevation.

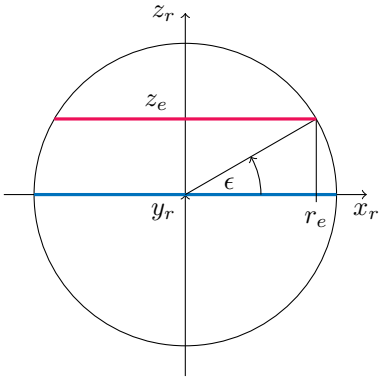


Figure 2: Illustration of z-coordinate and radius of an elevated circle on a sphere

The offset vector (see Fig. 1) is given by

$$\vec{o} = D(\alpha,\beta,0) \begin{pmatrix} 0 \\ 0 \\ z_e \end{pmatrix} = \begin{pmatrix} z_e\cos\alpha\sin\beta \\ z_e\sin\alpha\sin\beta \\ z_e\cos\beta \end{pmatrix} = \begin{pmatrix} r\,\sin\epsilon\cos\alpha\sin\beta \\ r\,\sin\epsilon\sin\alpha\sin\beta \\ r\,\sin\epsilon\cos\beta \end{pmatrix} \qquad (8)$$

and applied through `\tdplotsetrotatedcoordsorigin{`$\langle\vec{o}\rangle$`}` command of `tikz-3dplot`.

Drawing Circles of a Sphere

The parametric representation of a circle of a sphere in the rotated and offset coordinate system is

$$\begin{pmatrix} x(\varphi) \\ y(\varphi) \\ z(\varphi) \end{pmatrix} = \begin{pmatrix} r_e \cos\varphi \\ r_e \sin\varphi \\ 0 \end{pmatrix}. \tag{9}$$

After applying the coordinate transforms described above, we could just draw this circle by

$$\texttt{\textbackslash draw (0,0) circle } \langle r_e \rangle\texttt{;}$$

However, as we want to visualize the parts of the circle lying on the front and back sides of the sphere, we consider the parametric representation in the rotated but not *not*(!) offset coordinate system

$$\begin{pmatrix} x(\varphi) \\ y(\varphi) \\ z(\varphi) \end{pmatrix} = \begin{pmatrix} r_e \cos\varphi \\ r_e \sin\varphi \\ z_e \end{pmatrix}. \tag{10}$$

The respective screen coordinates are

$$\begin{aligned} \begin{pmatrix} x'(\varphi) \\ y'(\varphi) \\ z'(\varphi) \end{pmatrix} &= A \begin{pmatrix} x(\varphi) \\ y(\varphi) \\ z(\varphi) \end{pmatrix} = \begin{pmatrix} a_{xx} & a_{xy} & a_{xz} \\ a_{yx} & a_{yy} & a_{yz} \\ a_{zx} & a_{zy} & a_{zz} \end{pmatrix} \begin{pmatrix} r\cos\epsilon\cos\varphi \\ r\cos\epsilon\sin\varphi \\ r\sin\epsilon \end{pmatrix} \\ &= \begin{pmatrix} a_{xx}\cdot r\cos\epsilon\cos\varphi + a_{xy}\cdot r\cos\epsilon\sin\varphi + a_{xz}\cdot r\sin\epsilon \\ a_{yx}\cdot r\cos\epsilon\cos\varphi + a_{yy}\cdot r\cos\epsilon\sin\varphi + a_{yz}\cdot r\sin\epsilon \\ a_{zx}\cdot r\cos\epsilon\cos\varphi + a_{zy}\cdot r\cos\epsilon\sin\varphi + a_{zz}\cdot r\sin\epsilon \end{pmatrix}. \end{aligned} \tag{11}$$

By examining the $z'(\varphi)$ coordinate we can determine which parts of the circle are

$$\begin{aligned} &\text{on the front side (or “above” the drawing paper)} \quad z'(\varphi) > 0 \quad \text{and} \\ &\text{on the back side (or “below” the drawing paper)} \quad z'(\varphi) < 0 \end{aligned} \tag{12}$$

of the sphere. We denote by φ_0 the crossing angles between the front and back sides. In order to determine them we solve

$$0 \stackrel{!}{=} z'(\varphi_0) = a_{zx}\cdot r\cos\epsilon\cos\varphi_0 + a_{zy}\cdot r\cos\epsilon\sin\varphi_0 + a_{zz}\cdot r\sin\epsilon. \tag{13}$$

I must admit that I was too lazy to puzzle this out myself...;-) Matlab says:

$$\tan\left(\frac{\varphi_0}{2}\right) = \frac{a_{zy}\cos\epsilon \pm \sqrt{a_{zx}^2\cos^2\epsilon + a_{zy}^2\cos^2\epsilon - a_{zz}^2\sin^2\epsilon}}{a_{zx}\cos\epsilon - a_{zz}\sin\epsilon} \tag{14}$$

$$= \frac{a_{zy} \pm \sqrt{a_{zx}^2 + a_{zy}^2 - a_{zz}^2\tan^2\epsilon}}{a_{zx} - a_{zz}\tan\epsilon}, \tag{15}$$

where

$$a_{zz}^2\sin^2\epsilon \geq (a_{zx}^2 + a_{zy}^2)\cos^2\epsilon \quad \rightsquigarrow \quad \tan^2\epsilon \geq \frac{a_{zx}^2 + a_{zy}^2}{a_{zz}^2} \tag{16}$$

must hold. With the substitutions

$$u = a_{zy}, \tag{17}$$

$$v = \sqrt{a_{zx}^2 + a_{zy}^2 - a_{zz}^2\tan^2\epsilon} \quad \text{and} \tag{18}$$

$$w = a_{zx} - a_{zz}\tan\epsilon \tag{19}$$

we get

$$\tan\left(\frac{\varphi_0}{2}\right) = \frac{u \pm v}{w} \quad \rightsquigarrow \quad \varphi_0 = \begin{cases} 2\,\mathrm{arctan2}(u+v, w) \\ 2\,\mathrm{arctan2}(u-v, w) \end{cases} \tag{20}$$

Here we used the arctan2(x, y) function which is defined as

$$\operatorname{arctan2}(x,y) = \begin{cases} \arctan\left(\frac{x}{y}\right) & y > 0 \\ \arctan\left(\frac{x}{y}\right) + \pi & y < 0, x \geq 0 \\ \arctan\left(\frac{x}{y}\right) - \pi & y < 0, x < 0 \\ \frac{\pi}{2} & y = 0, x > 0 \\ -\frac{\pi}{2} & y = 0, x < 0 \\ 0 & y = 0, x = 0 \end{cases} \tag{21}$$

Iff condition (16) holds, Eqn. (13) has exactly two solutions,[3]

$$\varphi_{0,\mathrm{bf}} : \text{angle of back to front side crossing and}$$
$$\varphi_{0,\mathrm{fb}} : \text{angle of front to back side crossing,}$$

As PGF's `atan2` function takes values in $[-360°, +360°]$, we unwrap $\varphi_{0,\mathrm{bf}}$ and $\varphi_{0,\mathrm{fb}}$ as follows:

$$\text{if} \quad \varphi_{0,\mathrm{fb}} - \varphi_{0,\mathrm{bf}} > 360° \quad \text{then} \quad \varphi_{0,\mathrm{bf}} \leftarrow \varphi_{0,\mathrm{bf}} + 360°, \tag{22}$$
$$\text{if} \quad \varphi_{0,\mathrm{bf}} > \varphi_{0,\mathrm{fb}} \quad \text{then} \quad \varphi_{0,\mathrm{bf}} \leftarrow \varphi_{0,\mathrm{bf}} - 360°. \tag{23}$$

Now we can draw the arc on the sphere's back side in the rotated *and* offset coordinate system by

$$\texttt{\textbackslash draw (}\langle\varphi_{0,\mathtt{fb}}\rangle\texttt{:}\langle r_e\rangle\texttt{) arc (}\langle\varphi_{0,\mathtt{fb}}\rangle\texttt{:}\langle\varphi_{0,\mathtt{bf}} + 360°\rangle\texttt{:}\langle r_e\rangle\texttt{);}$$

and the arc on the sphere's front side by

$$\texttt{\textbackslash draw (}\langle\varphi_{0,\mathtt{bf}}\rangle\texttt{:}\langle r_e\rangle\texttt{) arc (}\langle\varphi_{0,\mathtt{bf}}\rangle\texttt{:}\langle\varphi_{0,\mathtt{fb}}\rangle\texttt{:}\langle r_e\rangle\texttt{);}$$

Otherwise, iff condition (16) does not hold, Eqn. (13) has no solutions, which means that the circle lies entirely either on the front side or on the back side of the sphere. In order to determine which is the case, we test if an arbitrary point on the circle lies above or below the drawing paper (cf. Eq. (12)). We choose $\varphi_0 = 0$ and evaluate the right side of Eq. (13)

$$a_{zx} \cdot r\cos\epsilon + a_{zz} \cdot r\sin\epsilon \begin{cases} \geq 0: & \text{front side circle,} \\ < 0: & \text{back side circle.} \end{cases} \tag{24}$$

Depending on the result, we draw the circle

$$\texttt{\textbackslash draw (0,0) circle (}\langle r_e\rangle\texttt{);}$$

in the front side or back side style.

3.2 The Package Source Code

```
%% == LaTeX PACKAGE tikz-3dplot-circleofsphere ================================
%%    Drawing circles of a sphere with tikz-3dplot
%%
%% Matthias Wolff, BTU Cottbus-Sentenberg
%% July 27, 2018
%%
%% References:
%% [1] J. Hein. The tikz-3dplot package. 2012. Online, retrieved July 20, 2018.
%%     http://mirror.ctan.org/graphics/pgf/contrib/tikz-3dplot/tikz-3dplot_documentation.pdf
%% [2] T. Tantau. TikZ & PGF - Manual for Version 3.0.1a. 2015. Online, retrieved July 22, 2018.
%%     http://mirror.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf
%% [3] Drawing Great Circles
%%     https://tex.stackexchange.com/questions/168521/spherical-triangles-and-great-circles

%% == REQUIRED PACKAGES ========================================================

```

[3] which coincide iff the left and right sides of condition (16) are equal

```
\RequirePackage{xifthen}
\RequirePackage{tikz}
\RequirePackage{tikz-3dplot}

%% == TikZ STYLES ============================================================

\tikzset{
  tdplotCsFront/.style={solid},
  tdplotCsBack/.style={dashed},
  tdplotCsFill/.style={opacity=0},
  tdplotPtFront/.style={},
  tdplotPtBack/.style={},
  tdplotCsDrawAux/.style={}
}

%% == COMMANDS ===============================================================

\newcommand{\tdplotCsComputeTransformRotScreen}{%
  % Computes the elements of the full rotation matrix
  %
  %   A = [\axx \axy \axz]
  %       [\ayx \ayy \ayz]
  %       [\azx \azy \azz].
  %
  % Ouput:
  %   \axx - Element A(1,1) of rotation matrix
  %   \axy - Element A(1,2) of rotation matrix
  %   ...
  %   \azz - Element A(3,3) of rotation matrix
  %
  \let\a\tdplotalpha
  \let\b\tdplotbeta
  \let\p\tdplotmainphi
  \let\t\tdplotmaintheta
  % Row 1: [\axx \axy \axz]
  \pgfmathsetmacro\axx{cos(\a)*cos(\b)*cos(\p) + cos(\b)*sin(\a)*sin(\p)}
  \pgfmathsetmacro\axy{cos(\a)*sin(\p) - cos(\p)*sin(\a)}
  \pgfmathsetmacro\axz{cos(\a)*cos(\p)*sin(\b) + sin(\a)*sin(\b)*sin(\p)}
  % Row 2: [\ayx \ayy \ayz]
  \pgfmathsetmacro\ayx{cos(\b)*cos(\p)*sin(\a)*cos(\t) - cos(\a)*cos(\b)*cos(\t)*sin(\p) - sin(\b)*sin(\t)}
  \pgfmathsetmacro\ayy{cos(\a)*cos(\p)*cos(\t) + sin(\a)*cos(\t)*sin(\p)}
  \pgfmathsetmacro\ayz{cos(\b)*sin(\t) - cos(\a)*sin(\b)*cos(\t)*sin(\p) + cos(\p)*sin(\a)*sin(\b)*cos(\t)}
  % Row 3: [\azx \azy \azz]
  \pgfmathsetmacro\azx{cos(\a)*cos(\b)*sin(\p)*sin(\t) - sin(\b)*cos(\t) - cos(\b)*cos(\p)*sin(\a)*sin(\t)}
  \pgfmathsetmacro\azy{-cos(\a)*cos(\p)*sin(\t) - sin(\a)*sin(\p)*sin(\t)}
  \pgfmathsetmacro\azz{cos(\b)*cos(\t) + cos(\a)*sin(\b)*sin(\p)*sin(\t) - cos(\p)*sin(\a)*sin(\b)*sin(\t)}
}

% ----------------------------------------------------------------------------

\newcommand{\tdplotCsDrawCircle}[5][]{%
  % Draws a circle of a sphere.
  %
  % Input:
  %   #1 - TikZ style
  %        - use tdplotCsFront/.style={...} to style the front side arc
  %        - use tdplotCsBack/.style={...} to style the back side arc
  %        - use tdplotCsFill/.style={...} to style the circle filling
  %        - use tdplotCsDrawAux to draw some auxiliary information
  %   #2 - Radius of sphere
  %   #3 - Azimuthal angle of drawing plane 1)
  %   #4 - Polar angle of drawing plane 2)
  %   #5 - Elevation angle of circle above the drawing plane. Permissible
  %        values are -90 < #5 < 90. Use 0 for drawing a great circle.
  %
  % Ouput:
  %   none
  %
  % Footnotes:
  %   1) passed as alpha to \tdplotsetrotatedcoords{alpha}{beta}{gamma}
  %   2) passed as beta to \tdplotsetrotatedcoords{alpha}{beta}{gamma}
  \begin{scope}[#1]                                                    % Macro scope >>
```

```
    % Do some computation                                                    #   ---------------------------------
    \pgfmathsetmacro\r  {#2}                                                %   Parse radius
    \pgfmathsetmacro\alp{#3}                                                %   Parse azimuthal angle (alpha)
    \pgfmathsetmacro\bet{#4}                                                %   Parse polar angle (beta)
    \pgfmathsetmacro\eps{#5}                                                %   Parse elevation angle (epsilon)
    \pgfmathsetmacro\re {\r*cos(\eps)}                                      %   Radius of circle
    \pgfmathsetmacro\ze {\r*sin(\eps)}                                      %   z-coordinate of drawing plane
    \pgfmathsetmacro\coX{\ze*cos(\alp)*sin(\bet)}                           %   x-coordinate offset for ze
    \pgfmathsetmacro\coY{\ze*sin(\alp)*sin(\bet)}                           %   y-coordinate offset for ze
    \pgfmathsetmacro\coZ{\ze*cos(\bet)}                                     %   z-coordinate offset for ze
    \coordinate (coffs) at (\coX,\coY,\coZ);                                %   Offset as coordinate value
    % Rotate and offset coordinate system                                   %   ---------------------------------
    \tdplotsetrotatedcoords{\alp}{\bet}{0}                                  %   Rotate coordinate system
    \tdplotsetrotatedcoordsorigin{(coffs)}                                  %   Offset coordinate system
    % Draw                                                                  %   ---------------------------------
    \begin{scope}[tdplot_rotated_coords]                                    %   Drawing scope >>
      \tdplotCsComputeTransformRotScreen                                    %     Compute full rotation matrix
      \pgfmathsetmacro\tanEps{tan(\eps)}                                    %     Tangent of elevation angle
      \pgfmathsetmacro\bOneside{((\tanEps)^2)>=(((\azx)^2+(\azy)^2)/(\azz)^2)} %   Circle entirely on one side?
      \ifthenelse{\isin{tdplotCsFill}{#1}}{                                 %     Fill style passed >>
        \fill[tdplotCsFill] (0,0) circle (\re);                             %       Draw filling of circle
      }{}                                                                   %     <<
      \ifthenelse{\bOneside=1}{                                             %     Circle on one side of sphere >>
        \pgfmathsetmacro\bFrontside{(\azx*\re+\azz*\ze)>=0}                 %       Circle entirely on front side?
        \ifthenelse{\bFrontside=1}                                          %       |
          {\draw[tdplotCsFront] (0,0) circle (\re);}                        %       Draw on front side
          {\draw[tdplotCsBack]  (0,0) circle (\re);}                        %       Draw on back side
      }{                                                                    %     << Circle on both sides >>
        \pgfmathsetmacro\u{\azy}                                            %       Substitution u=...
        \pgfmathsetmacro\v{sqrt( (\azx)^2 + (\azy)^2 - (\azz)^2*(\tanEps)^2 )} %    Substitution v=...
        \pgfmathsetmacro\w{\azx - \azz*\tanEps}                             %       Substitution w=...
        \pgfmathsetmacro\phiBf{2*atan2(\u-\v,\w)}                           %       Back->front crossing angle
        \pgfmathsetmacro\phiFb{2*atan2(\u+\v,\w)}                           %       Front->back crossing angle
        \pgfmathsetmacro\bUnwrapA{(\phiFb-\phiBf)>360}                      %       Unwrap front->back angle #1?
        \pgfmathsetmacro\bUnwrapB{\phiBf>\phiFb}                            %       Unwrap front->back angle #2?
        \ifthenelse{\bUnwrapA=1}{\pgfmathsetmacro\phiBf{\phiBf+360}}{}      %       Unwrap front->back angle #1
        \ifthenelse{\bUnwrapB=1}{\pgfmathsetmacro\phiBf{\phiBf-360}}{}      %       Unwrap front->back angle #2
        \draw[tdplotCsBack]  (\phiFb:\re) arc (\phiFb:{\phiBf+360}:\re);    %       Draw back side arc
        \draw[tdplotCsFront] (\phiBf:\re) arc (\phiBf:\phiFb:\re);          %       Draw back side arc
      }                                                                     %     <<
      % Auxliliary drawing (for debugging and illustration)                 %     - - - - - - - - - - - - - - - - - -
      \ifthenelse{\isin{tdplotCsDrawAux}{#1}}{                              %     Auxiliary drawing activated >>
        \draw[red!40,->] (-\re,0,0) -- (\re,0,0) node[anchor=north] {$x_d$}; %       x-axis of drawing corrd. system
        \draw[red!40,->] (0,-\re,0) -- (0,\re,0) node[anchor=north] {$y_d$}; %       y-axis of drawing corrd. system
        \draw[red!40,->] (0,0,0)    -- (0,0,\re) node[anchor=north] {$z_d$}; %       z-axis of drawing corrd. system
        \ifthenelse{\bOneside=0}{                                           %       Circ.on both sides of sphere >>
          \node[red] at (\phiBf:\re) {$\circ$};                             %         Indicate back-front crossing
          \node[red] at (\phiFb:\re) {$\times$};                            %         Indicate front-back crossing
        }{}                                                                 %       <<
        \coordinate (coffs) at (-\coX,-\coY,-\coZ);                         %       HACK: Forcibly reset ...
        \tdplotsetrotatedcoordsorigin{(coffs)}                              %       ... coordinate system
        \begin{scope}[tdplot_rotated_coords]                                %       Aux. display scope >>
          \node[tdplot_screen_coords,red,anchor=north west] at (0.7*\r,-0.9*\r) %     Make a litte display ...
            {\parbox{200pt}{\footnotesize                                   %       ... >>
              $\theta=\tdplotmaintheta^\circ, \phi=\tdplotmainphi^\circ$\\  %         Main coord. sys. parameters
              $\alpha=\alp^\circ, \beta=\bet^\circ,                         %         Rot. coord. sys. parameters
               \epsilon\!=\!\eps^\circ\!$\\                                 %         Drawing plane elev. angle
              $a_{zx}=\azx, a_{zy}=\azy, a_{zz}=\azz$\\                     %         Elems. of full rot. matrix
              $r_e\!=\!\re, z_e\!=\!\ze$\\                                  %         Radius and z-elevation
              $\texttt{\textbackslash bOneside}\!=\!\bOneside$,             %         One-side circle flag
              \ifthenelse{\bOneside=1}{                                     %         One-side circle >>
                $\texttt{\textbackslash bFrontside}\!=\!\bFrontside$\\      %           Front-side flag
              }{                                                            %         << Two-side circle >>
                $\texttt{\textbackslash bUnwrapA}\!=\!\bUnwrapA$,           %           Angle unwrap flag #1
                $\texttt{\textbackslash bUnwrapB}\!=\!\bUnwrapB$\\          %           Angle unwrap flag #2
                $\circ\!: \!\texttt{\textbackslash phiBf}\!=\!\phiBf^\circ\!, %        Back-front crossing angle
                 \times\!:\!\texttt{\textbackslash phiFb}\!=\!\phiFb^\circ$\\ %       Front-back crossing angle
              }                                                             %         <<
            }};                                                             %       <<
        \end{scope}                                                         %     << (Aux. display scope)
      }{}                                                                   %   << (Auxiliary drawing activated)
    \end{scope}                                                             % << (Drawing scope)
```

```
  \end{scope}                                                    % << (Macro scope)
}

% ---------------------------------------------------------------------------

\newcommand{\tdplotCsDrawGreatCircle}[4][]{%
  % Draws a great circle.
  %
  % Input:
  %   #1 - TikZ style
  %        - use tdplotCsFront/.style={...} to style the front side arc
  %        - use tdplotCsBack/.style={...} to style the back side arc
  %        - use tdplotCsFill/.style={...} to style the circle filling
  %        - use tdplotCsDrawAux to draw some auxiliary information
  %   #2 - Radius of sphere
  %   #3 - Azimuthal angle of drawing plane 1)
  %   #4 - Polar angle of drawing plane 2)
  %
  % Ouput:
  %   none
  %
  % Footnotes:
  %   1) passed as alpha to \tdplotsetrotatedcoords{alpha}{beta}{gamma}
  %   2) passed as beta to \tdplotsetrotatedcoords{alpha}{beta}{gamma}
  \tdplotCsDrawCircle[#1]{#2}{#3}{#4}{0}
}

% ---------------------------------------------------------------------------

\newcommand{\tdplotCsDrawLatCircle}[3][]{%
  % Draws a circle of latitude.
  %
  % Input:
  %   #1 - TikZ style
  %        - use tdplotCsFront/.style={...} to style the front side arc
  %        - use tdplotCsBack/.style={...} to style the back side arc
  %        - use tdplotCsFill/.style={...} to style the circle filling
  %        - use tdplotCsDrawAux to draw some auxiliary information
  %   #2 - Radius of sphere
  %   #3 - Elevation angle of circle above the drawing plane. Permissible
  %        values are -90 < #5 < 90. Use 0 for drawing a great circle.
  %
  % Ouput:
  %   none
  \tdplotCsDrawCircle[#1]{#2}{0}{0}{#3}
}

% ---------------------------------------------------------------------------

\newcommand{\tdplotCsDrawLonCircle}[3][]{%
  % Draws a circle of longitude.
  %
  % Input:
  %   #1 - TikZ style
  %        - use tdplotCsFront/.style={...} to style the front side arc
  %        - use tdplotCsBack/.style={...} to style the back side arc
  %        - use tdplotCsFill/.style={...} to style the circle filling
  %        - use tdplotCsDrawAux to draw some auxiliary information
  %   #2 - Radius of sphere
  %   #3 - Azimuthal angle of drawing plane 1)
  %
  % Ouput:
  %   none
  %
  % Footnotes:
  %   1) passed as alpha to \tdplotsetrotatedcoords{alpha}{beta}{gamma}
  \tdplotCsDrawCircle[#1]{#2}{#3}{90}{0}
}

% ---------------------------------------------------------------------------

\newcommand{\tdplotCsFrontsidePoint}{%
```

```
  % Invoked by \tdplotCsDrawPoint to draw a point on the front side of a sphere.
  % Redefine to customize.
  \textbullet%
}

% ------------------------------------------------------------------------

\newcommand{\tdtlotCsBacksidePoint}{%
  % Invoked by \tdplotCsDrawPoint to draw a point on the back side of a sphere.
  % Redefine to customize.
  $\circ$%
}

% ------------------------------------------------------------------------

\newcommand{\tdplotCsDrawPoint}[4][]{%
  % Draws a point on a sphere.
  %
  % Input:
  %   #1 - TikZ style
  %        - use tdplotPtFront/.style={...} to style a front side point
  %        - use tdplotPtBack/.style={...} to style a back side  point
  %   #2 - Radius of sphere
  %   #3 - Azimuthal angle of drawing plane 1)
  %   #4 - Polar angle of drawing plane 2)
  %
  % Ouput:
  %   none
  %
  % Remarks:
  %   - Redefine \tdplotCsFrontsidePoint to customize drawing of a front side
  %     point.
  %   - Redefine \tdplotCsBacksidePoint to customize drawing of a back side
  %     point.
  %
  % Footnotes:
  %   1) passed as alpha to \tdplotsetrotatedcoords{alpha}{beta}{gamma}
  %   2) passed as beta to \tdplotsetrotatedcoords{alpha}{beta}{gamma}
  \begin{scope}[#1]                                                    % Macro scope >>
    \pgfmathsetmacro{\r}{#2}                                           %   Parse radius
    \pgfmathsetmacro{\alp}{#3}                                         %   Parse alpha angle
    \pgfmathsetmacro{\bet}{#4}                                         %   Parse beta angle
    \tdplotsetrotatedcoords{\alp}{\bet}{0}                             %   Set rotated coord. system
    \begin{scope}[tdplot_rotated_coords]                               %   Draw in rotated coord. system >>
      \tdplotCsComputeTransformRotScreen                               %     Get \azz
      \pgfmathsetmacro{\bVisible}{\azz>0}                              %     Test if point is on visible side
       \ifthenelse{\bVisible=1}{%                                      %     Point on front side >>
         \node[tdplotPtFront] at (0,0,\r) {\tdplotCsFrontsidePoint};   %       Draw it
       }{%                                                             %     << Point on back side >>
         \node[tdplotPtBack] at (0,0,\r) {\tdplotCsBacksidePoint};     %       Draw it
       }                                                               %     <<
    \end{scope}                                                        %   <<
  \end{scope}                                                          % <<
}

%% == EOF ==============================================================
```

3.3 An Auxiliary Matlab Script

```
%% == LaTeX PACKAGE tikz-3dplot-circleofsphere ================================
%     Drawing circles of a sphere with tikz-3dplot
%
%  Matthias Wolff, BTU Cottbus-Sentenberg
%  July 26, 2018
%
%  References:
%  [1] J. Hein. The tikz-3dplot package. 2012. Online, retrieved July 20, 2018.
%      https://mirror.hmc.edu/ctan/graphics/pgf/contrib/tikz-3dplot/tikz-3dplot_documentation.pdf
%

%% Rotation matrices ===========================================================
syms a b p t
```

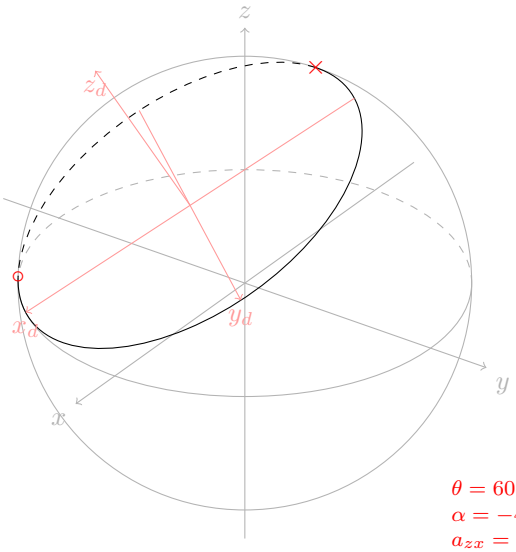
```
14
15 % R rotation matrix -----------------------------------------------------------
16 Rz = [  cos(p) -sin(p)   0
17        sin(p)  cos(p)   0
18        0       0        1      ];
19
20 Rx = [  1       0        0
21        0       cos(t)  -sin(t)
22        0       sin(t)   cos(t) ];
23
24 % - [1] eq. (2.1) line 2
25 % R = Rz*Rx; disp(R);
26
27 % - [1] eq. (2.1) line 3
28 % R = [  cos(p)         sin(p)         0
29 %       -cos(t)*sin(p)  cos(t)*cos(p) -sin(t)
30 %        sin(t)*sin(p) -sin(t)*cos(p)  cos(t) ];
31
32 % - [1] eq. (2.1) line 3, corrected
33 R = (Rz*Rx).';
34
35 % -- D rotation matrix ------------------------------------------------------
36 Dz = [  cos(a) -sin(a)  0
37        sin(a)  cos(a)  0
38        0       0       1      ];
39
40 Dy = [  cos(b)  0       sin(b)
41        0       1       0
42       -sin(b)  0       cos(b) ];
43
44 Dx = [  1       0       0
45        0       cos(b) -sin(b)
46        0       sin(b)  cos(b) ];
47
48 D = Dz*Dy; disp(D);
49
50 % -- Full rotation matrix ---------------------------------------------------
51 A = R*D; disp(A);
52 axx = A(1,1); axy = A(1,2); axz = A(1,3);
53 ayx = A(2,1); ayy = A(2,2); ayz = A(2,3);
54 azx = A(3,1); azy = A(3,2); azz = A(3,3);
55
56 %% == Transform a vector (world -> screen) ===================================
57 syms x y z
58 p = [ x
59       y
60       z ];
61 q=A*p;
62 disp(q);
63
64 %% == View angle ==============================================================
65 syms p0 r eps azx azy azz
66 assume(p0,'real');
67 assume(r,'real');
68 assume(eps,'real');
69 assume(azx,'real');
70 assume(azy,'real');
71 assume(azz,'real');
72 eqn = azx*r*cos(eps)*cos(p0) + azy*r*cos(eps)*sin(p0) + azz*r*sin(eps) == 0
73 solve(eqn,p0,'Real',true)
74
75 %% == EOF =====================================================================
```

References

[1] Jeff Hein. The `tikz-3dplot` package. `http://mirror.ctan.org/graphics/pgf/contrib/tikz-3dplot/tikz-3dplot_documentation.pdf`, 2012. Retrieved: July 27, 2018.

[2] Till Tantau. Tikz & pgf - manual for version 3.0.1a. `http://mirror.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf`, 2015. Retrieved: July 27, 2018.

[3] Matthias Wolff. The `tikz-3dplot-circleofsphere` package: Drawing circles of a sphere with `tikz-3dplot`. `https://github.com/matthias-wolff/tikz-3dplot-circleofsphere`, 2018. Retrieved: July 27, 2018.



$\theta = 60.0°, \phi = 125.0°$
$\alpha = -40.0°, \beta = 30°, \epsilon = 30°$
$a_{zx} = -0.05588, a_{zy} = 0.8365, a_{zz} = 0.54507$
$r_e = 2.59808, z_e = 1.5$
$\texttt{\textbackslash bOneside} = 0,\ \texttt{\textbackslash bUnwrapA} = 0,\ \texttt{\textbackslash bUnwrapB} = 1$
$\circ : \texttt{\textbackslash phiBf} = -18.22858°, \times : \texttt{\textbackslash phiFb} = 205.86197°$

```
\documentclass{standalone}
\usepackage[dvipsnames]{xcolor}
\usepackage{tikz-3dplot-circleofsphere}

\begin{document}

  \def\elev{  30} \pgfmathsetmacro{\tdpTheta}{90-\elev}
  \def\azim{  35} \pgfmathsetmacro{\tdpPhi}{90+\azim}
  \def\R{3}
  \tdplotsetmaincoords{\tdpTheta}{\tdpPhi}
  \begin{tikzpicture}[scale=1,tdplot_main_coords]
    \begin{scope}[black!30,name=auxiliary]
      \draw[tdplot_screen_coords] (0,0,0) circle (\R);
      \draw[->] (-1.3*\R,0,0) -- (1.3*\R,0,0) node[anchor=north east]{$x$};
      \draw[->] (0,-1.3*\R,0) -- (0,1.3*\R,0) node[anchor=north west]{$y$};
      \draw[->] (0,0,-1.3*\R) -- (0,0,1.3*\R) node[anchor=south]{$z$};
      \tdplotCsDrawCircle{\R}{0}{0}{0};
    \end{scope}
    \begin{scope}
%     \tdplotCsDrawLatCircle[tdplotCsDrawAux]{\R}{-30}
      % --
      \tdplotCsDrawCircle[tdplotCsDrawAux]{\R}{-40}{30}{30}
      % --
%       \foreach \a in {0,15,...,345}
%         { \tdplotCsDrawCircle[very thin,gray]{\R}{\a}{90}{0} }
%       \foreach \a in {-75,-60,...,75}
%         { \tdplotCsDrawCircle[very thin,gray]{\R}{0}{0}{\a} }
      % -- Pathologic cases -->
%       \tdplotCsDrawCircle{\R}{35}{60}{0}
      % <--
    \end{scope}
  \end{tikzpicture}

\end{document}
```