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

Matthias Wolff[0000–0002–3895–7313]

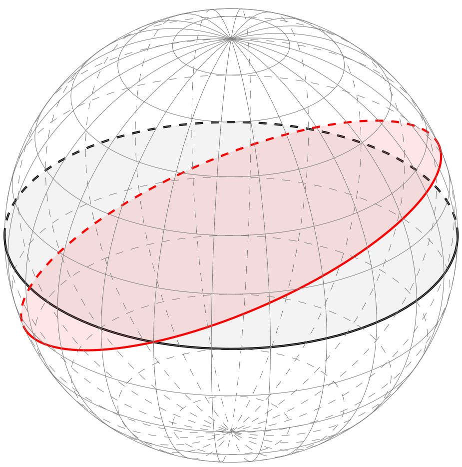
BTU Cottbus-Senftenberg

July 30, 2018

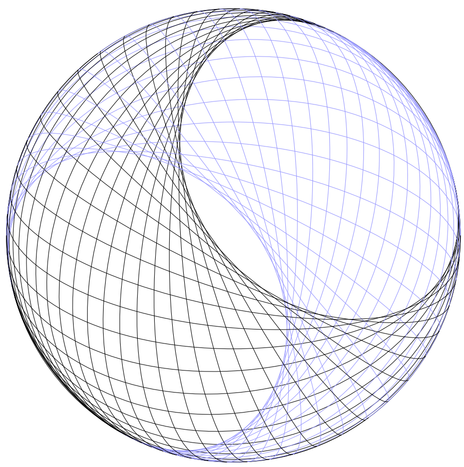
Abstract

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.

Note: Package and documentation are under construction!



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

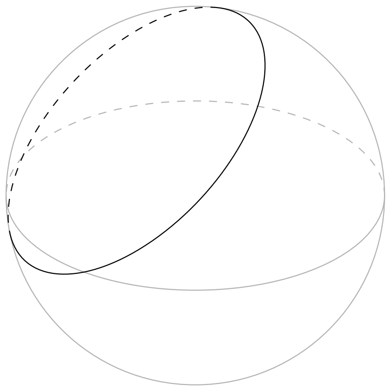


```
\documentclass{standalone}
\usepackage{tikz-3dplot-circleofsphere}
\begin{document}
  \centering
  \def\R{3}
  \tdplotsetmaincoords{60}{125}
  \begin{tikzpicture}[tdplot_main_coords]
    \def\e{80};
    \draw[tdplot_screen_coords,very thin] (0,0,0) circle (\R);
    \foreach \a in {0,5,...,175} {
      \tdplotCsDrawGreatCircle%
        [very thin, tdplotCsBack/.style={very 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}[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}`. That's all.

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–

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

Draws a great circle.
Equivalent to `\tdplotCsDrawCircleOfSphere[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–

`\tdplotCsDrawLatCircle[style]{r}{epsilon}`

Draws a circle of latitude.
Equivalent to `\tdplotCsDrawCircleOfSphere[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 a great circle.

Output

–none–

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

Draws a circle of longitude.
Equivalent to `\tdplotCsDrawCircleOfSphere[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–

`\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 • use `tdplotPtDrawAux` 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–

Remarks

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

2.3 Geographic Drawing Commands

`\tdplotCsDrawCircleLL[style]r}{lat}{lon}{elev}`

[TODO: ...]

`\tdplotCsDrawLatitudeCircleLL[style]r}{lat}`

[TODO: ...]

`\tdplotCsDrawLongitudeCircleLL[style]r}{lon}`

[TODO: ...]

`\tdplotCsDrawPointLL[style]{r}{lat}{lon}`

[TODO: ...]

2.4 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`.

2.5 Examples

Examples ?? and ?? (see below) demonstrate the usage of the `tikz-3dplot-circleofsphere` package.
[**TODO:** Fix examples!]

2.6 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

[**TODO:** Briefly explain!]

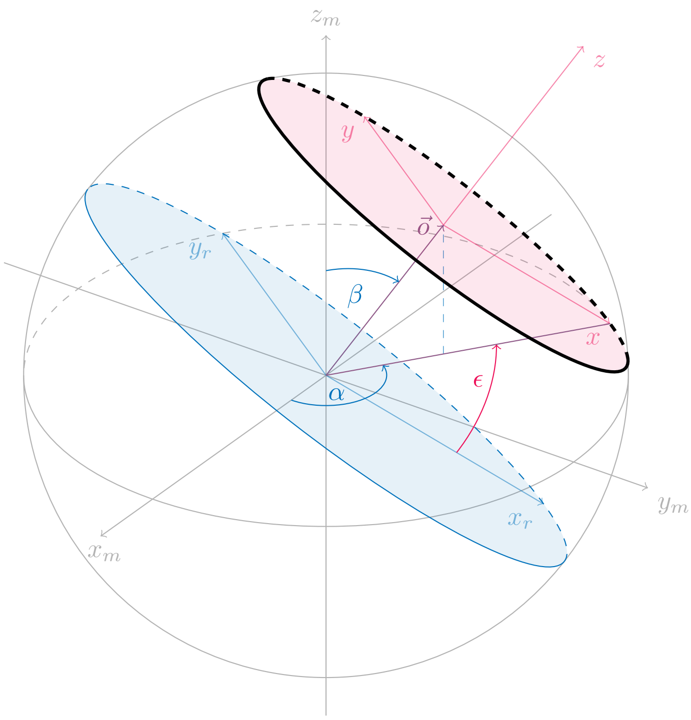


Figure 1: [**TODO:** ...]

Coordinate Transforms with `tikz-3dplot`

For drawing circles on a sphere, we use the `circle` and `arc` path construction operations of TikZ. As TikZ will only draw circles and arcs on the xy-plane, we need to rotate and possibly offset the coordinate system for drawing circles of spheres. This functionality is provided by the `tikz-3dplot` [1] package.

First, `tikz-3dplot` provides a *main coordinate system* which is basicly defining the view point 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

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

with the rotation matrix[1]

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

Second, for drawing circles and arcs outside the xy-plane, we need to rotate the coordinate system

[1] Equation (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 on opposite order and direction.

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])

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

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

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

With the coordinate transforms described so far, we can only draw circles and arcs whose center is the origin of the main coordinate systems. For drawing other circles on a sphere, we additionally need to offset the origin of the rotated coordinate system. This is provided by the `\tdplotsetrotatedcoordsorigin` command of `tikz-3dplot`.

[**TODO:** Describe how!]

Drawing Circles of a Sphere

The parametric representation of a circle at a plane parallel to the xy-plane is

$$\begin{pmatrix} x(\varphi) \\ y(\varphi) \\ z(\varphi) \end{pmatrix} = \begin{pmatrix} r_e\cos\varphi \\ r_e\sin\varphi \\ z_e \end{pmatrix}, \tag{6}$$

where $-180° < \varphi \leq 180°$ the angle parameter,

$$r_e = \cos\epsilon \tag{7}$$

the radius,

$$z_e = \sin\epsilon \tag{8}$$

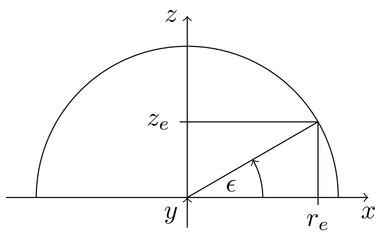


Figure 2: Illustration of z-coordinate and radius of an elevated circle on a sphere

the height above the xy-plane, and ϵ the elevation angle. Fig. 2 shows an illustration.
Note that we actually *draw* this circle in the rotated *and offset* coordinate system where it takes the form

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

However, we will stick to Eqn. (6) for simplicity. The screen coordinates for Eqn. (6) are

$$\begin{aligned} \begin{pmatrix} x'(\varphi) \\ y'(\varphi) \\ z'(\varphi) \end{pmatrix} = A \begin{pmatrix} x(\varphi) \\ y(\varphi) \\ z(\varphi) \end{pmatrix} &= \begin{pmatrix} a_{xx} & a_{xy} & a_{xz} \\ a_{yx} & a_{yy} & a_{yz} \\ a_{zx} & a_{zy} & a_{zz} \end{pmatrix} \begin{pmatrix} r\cos\epsilon\cos\varphi \\ r\cos\epsilon\sin\varphi \\ r\sin\epsilon \end{pmatrix} \\ &= \begin{pmatrix} a_{xx}\cdot r\cos\epsilon\cos\varphi + a_{xy}\cdot r\cos\epsilon\sin\varphi + a_{xz}\cdot r\sin\epsilon \\ a_{yx}\cdot r\cos\epsilon\cos\varphi + a_{yy}\cdot r\cos\epsilon\sin\varphi + a_{yz}\cdot r\sin\epsilon \\ a_{zx}\cdot r\cos\epsilon\cos\varphi + a_{zy}\cdot r\cos\epsilon\sin\varphi + a_{zz}\cdot r\sin\epsilon \end{pmatrix}. \end{aligned} \tag{10}$$

The $z'(\varphi)$ coordinates are not plotted. However, they are useful for determining which parts of the circle are

$$\begin{aligned} &\text{on the front side} \quad z'(\varphi) \geq 0 \quad \text{and} \\ &\text{on the back side} \quad z'(\varphi) < 0 \end{aligned} \tag{11}$$

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

$$0 \overset{!}{=} z'(\varphi_0) = a_{zx}\cdot r\cos\epsilon\cos\varphi_0 + a_{zy}\cdot r\cos\epsilon\sin\varphi_0 + a_{zz}\cdot r\sin\epsilon. \tag{12}$$

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

$$\tan\left(\frac{\varphi_0}{2}\right) = \frac{a_{zy}\cos\epsilon \pm \sqrt{a_{zx}^2\cos^2\epsilon + a_{zy}^2\cos^2\epsilon - a_{zz}^2\sin^2\epsilon}}{a_{zx}\cos\epsilon - a_{zz}\sin\epsilon} \tag{13}$$

$$= \frac{a_{zy} \pm \sqrt{a_{zx}^2 + a_{zy}^2 - a_{zz}^2\tan^2\epsilon}}{a_{zx} - a_{zz}\tan\epsilon}, \tag{14}$$

where

$$a_{zz}^2\sin^2\epsilon \geq (a_{zx}^2 + a_{zy}^2)\cos^2\epsilon \quad \rightsquigarrow \quad \tan^2\epsilon \geq \frac{a_{zx}^2 + a_{zy}^2}{a_{zz}^2} \tag{15}$$

must hold. With the substitutions

$$u = a_{zy}, \tag{16}$$

$$v = \sqrt{a_{zx}^2 + a_{zy}^2 - a_{zz}^2\tan^2\epsilon} \quad \text{and} \tag{17}$$

$$w = a_{zx} - a_{zz}\tan\epsilon \tag{18}$$

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

we get

$$\tan\left(\frac{\varphi_0}{2}\right) = \frac{u \pm v}{w} \quad \rightsquigarrow \quad \varphi_0 = \begin{cases} 2\arctan2(u+v, w) \\ 2\arctan2(u-v, w) \end{cases} \tag{19}$$

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

$$\arctan2(x, y) = \begin{cases} \arctan\left(\frac{x}{y}\right) & y > 0 \\ \arctan\left(\frac{x}{y}\right) + \pi & y < 0, x \geq 0 \\ \arctan\left(\frac{x}{y}\right) - \pi & y < 0, x < 0 \\ \frac{\pi}{2} & y = 0, x > 0 \\ -\frac{\pi}{2} & y = 0, x < 0 \\ 0 & y = 0, x = 0 \end{cases} \tag{20}$$

Iff condition (15) holds, Eqn. (12) has exactly two solutions,[3]

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

Otherwise it has no solutions, which means that the circle lies entirely either on the front side or on the back side of the sphere.

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 =======================================================

\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]
```

[3] which coincide iff the left and right sides of condition (15) are equal

```
  %       [\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{\tdplotCsDrawCircleOfSphere}[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}
  \tdplotCsDrawCircleOfSphere[#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
  \tdplotCsDrawCircleOfSphere[#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}
  \tdplotCsDrawCircleOfSphere[#1]{#2}{#3}{90}{0}
}

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

\newcommand{\tdplotgcFrontsidePoint}{%
  % 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
%        - use tdplotPtDrawAux 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
%
% 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 visible side >>
       \node[tdplotPtFront] at (0,0,\r) {\tdplotCsFrontsidePoint};       %       Draw it
     }{%                                                                 %     << Point on invisible 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

% R rotation matrix -----------------------------------------------------------
Rz = [  cos(p) -sin(p)   0
        sin(p)  cos(p)   0
        0       0        1      ];

Rx = [  1       0        0
        0       cos(t)  -sin(t)
        0       sin(t)   cos(t) ];

% - [1] eq. (2.1) line 2
% R = Rz*Rx; disp(R);

% - [1] eq. (2.1) line 3
% R = [  cos(p)         sin(p)         0
%       -cos(t)*sin(p)  cos(t)*cos(p) -sin(t)
%        sin(t)*sin(p) -sin(t)*cos(p)  cos(t) ];

% - [1] eq. (2.1) line 3, corrected
R = (Rz*Rx).';
```

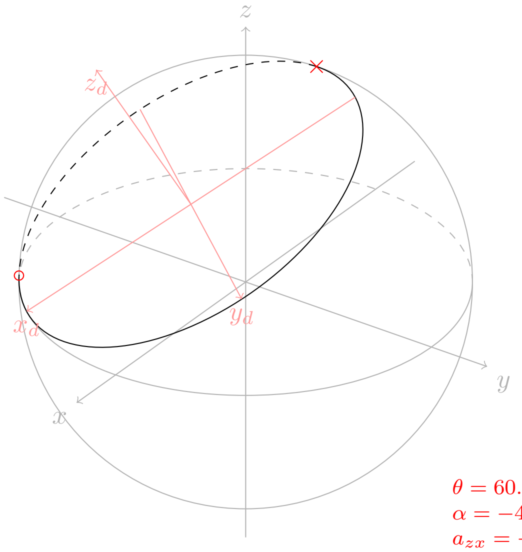
```

% -- D rotation matrix --------------------------------------------------------
Dz = [  cos(a) -sin(a)  0
        sin(a)  cos(a)  0
        0       0       1      ];

Dy = [  cos(b)  0       sin(b)
        0       1       0
       -sin(b)  0       cos(b) ];

Dx = [  1       0       0
        0       cos(b) -sin(b)
        0       sin(b)  cos(b) ];

D = Dz*Dy; disp(D);

% -- Full rotation matrix -----------------------------------------------------
A = R*D; disp(A);
axx = A(1,1); axy = A(1,2); axz = A(1,3);
ayx = A(2,1); ayy = A(2,2); ayz = A(2,3);
azx = A(3,1); azy = A(3,2); azz = A(3,3);

%% == Transform a vector (world -> screen) ====================================
syms x y z
p = [ x
      y
      z ];
q=A*p;
disp(q);

%% == View angle ==============================================================
syms p0 r eps azx azy azz
assume(p0,'real');
assume(r,'real');
assume(eps,'real');
assume(azx,'real');
assume(azy,'real');
assume(azz,'real');
eqn = azx*r*cos(eps)*cos(p0) + azy*r*cos(eps)*sin(p0) + azz*r*sin(eps) == 0
solve(eqn,p0,'Real',true)

% syms p0 u v w
% assume(p0,'real');
% assume(u,'real');
% assume(v,'real');
% assume(w,'real');
% eqn = u*cos(p0) + v*sin(p0) + w == 0;
% solve(eqn,p0,'Real',true)

%% == 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.



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

```latex
\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]{$z$};
      \tdplotCsDrawCircleOfSphere{\R}{0}{0}{0};
    \end{scope}
    \begin{scope}
%       \tdplotCsDrawLatCircle[tdplotCsDrawAux]{\R}{-30}
      % --
      \tdplotCsDrawCircleOfSphere[tdplotCsDrawAux]{\R}{-40}{30}{30}
      % --
%       \foreach \a in {0,15,...,345}
%         { \tdplotCsDrawCircleOfSphere[very thin,gray]{\R}{\a}{90}{0} }
%       \foreach \a in {-75,-60,...,75}
%         { \tdplotCsDrawCircleOfSphere[very thin,gray]{\R}{0}{0}{\a} }
      % -- Pathologic cases -->
%       \tdplotCsDrawCircleOfSphere{\R}{35}{60}{0}
      % <--
    \end{scope}
  \end{tikzpicture}

\end{document}
```