Oblique Spherical Triangles Toolbox

Introduction

Spherical trigonometry is generally much more involved than its plain counterpart. A nice example of this is the seemingly simple spherical triangle (a triangle on a sphere). The usual laws for plain triangles no longer hold: the sum of the angles can easily exceed 180°, the Pythagorean theorem does not apply when one of the angles is 90°, and so on.

The formulas to find quantities in spherical triangles are quite tedious. Their derivation is also not very trivial, which makes them also harder to remember. For that reason, this little toolbox has been created.

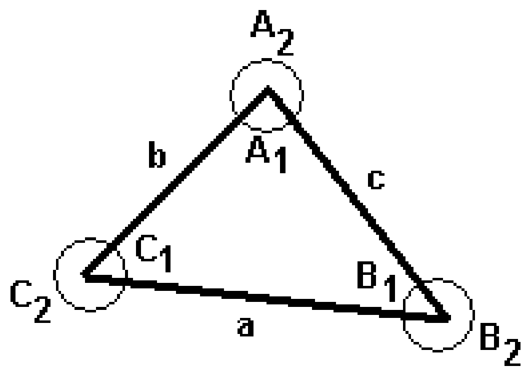

Figure 1: An arbitrary oblique spherical triangle.

A general spherical triangle is shown in Figure 1. Note that there are two angles associated with every "corner" of the triangle. Also keep in mind that each "side" of a spherical triangle is not a length, but an angle (with respect to the center of the sphere). Then, if 3 arbitrary quantities (angles, sides) are given and the other 3 quantities are to be found, 6 different sub-problems can be identified. These can be defined as follows:

Table 1: The 6 categories of oblique spherical triangles. Adapted from Table A-1 in [Wertz(2001)]

Type	**Given**	**Find**	**No. of Solutions**
Side-Side-Side	a, b, c	A, B, C	0 or 2
Side-Side-Angle	a, b, A	B, C, c	0 or 2
Side-Angle-Side	a, C, b	A, B, c	2
Angle-Angle-Side	A, B, a	b, c, C	0 or 2
Angle-Side-Angle	A, c, B	a, C, b	2
Angle-Angle-Angle	A, B, C	a, b, c	0 or 2

All these problems involve rather long equations which will not be repeated here. The interested reader is referred to Appendix A of [Wertz(2001)], or similar standard works on spherical trigonometry.

[Wertz(2001)] uses a somewhat unconventional method to automate solving these problems, namely the `acos2`-function. Indeed it is the four-quadrant arccosine function, which is defined as

$$\mathrm{acos2}(x) = H(x) \cdot \mathrm{acos}(x),$$

where $H(x) = +1$ if $x < 180°$, and $H(x) = -1$ if $x > 180°$. Defined like this, the `acos2` function completely does away with the need of user intervention when solving either of the 6 sub-problems, thus enabling full automation.

The Toolbox

This little toolbox is simply an implementation of all of the above sub-problems. All functions are fully vectorized and should work for vector or matrix input. The main components are:

`sss.m` and `sssd.m`

Usage:

```
[A1, B1, C1, A2, B2, C2] = sss(a, b, c)
```

Returns both solutions to the Side-Side-Side problem. If no solution exists, `NaN` is returned. Uses the `acos2` function. `sss.m` uses radians, whereas `sssd.m` uses degrees.

`ssa.m` and `ssad.m`

Usage:

```
[B1, C1, c1, B2, C2, c2] = ssa(a, b, A)
```

Returns both solutions to the Side-Side-Angle problem. If no solution exists, `NaN` is returned. Uses the implementations of the Middle Angle Law (`mal.m`) and Middle Side Law (`msl.m`). `ssa.m` uses radians, whereas `ssad.m` uses degrees.

`sas.m` and `sasd.m`

Usage:

```
[c1, A1, B1, c2, A2, B2] = sas(a, C, b)
```

Returns both solutions to the Side-Angle-Side problem. Uses the `acos2` function. `sas.m` uses radians, whereas `sasd.m` uses degrees.

`aas.m` and `aasd.m`

Usage:

```
[b1, c1, C1, b2, c2, C2] = aas(A, B, a)
```

Returns both solutions to the Angle-Angle-Side problem. If no solution exists, `NaN` is returned. Uses the implementations of the Middle Angle Law (`mal.m`) and Middle Side Law (`msl.m`). `aas.m` uses radians, whereas `aasd.m` uses degrees.

`asa.m` and `asad.m`

Usage:

```
[C1, a1, b1, C2, a2, b2] = asa(A, B, c)
```

Returns both solutions to the Angle-Side-Angle problem. Uses the `acos2` function. `asa.m` uses radians, whereas `asad.m` uses degrees.

`aaa.m` and `aaad.m`

Usage:

```
[a1, b1, c1, a2, b2, c2] = aaa(A, B, C)
```

Returns both solutions to the Angle-Angle-Angle problem. If no solution exists, `NaN` is returned. Uses the `acos2` function. `aaa.m` uses radians, whereas `aaad.m` uses degrees.

Auxiliary Functions

These are included in the auxiliary folder. They are:

`mal.m` and `mald.m`

The implementation of the Middle-Angle Law for spherical triangles:

$$\begin{aligned}
\sin c &= \frac{\sin a \cos b \cos B + \sin b \cos a \cos A}{1 - \sin a \sin b \sin A \sin B}, \\
\cos c &= \frac{\cos a \cos b + \sin a \sin b \cos A \cos B}{1 - \sin a \sin b \sin A \sin B}, \\
c &= \text{atan2}(\sin c,\, \cos c).
\end{aligned}$$

This function is used only internally. Instructions on how to use it manually are given in the help.

`msl.m` and `msld.m`

The implementation of the Middle-Side Law for spherical triangles:

$$\begin{aligned}
\sin C &= \frac{\sin A \cos B \cos b + \sin B \cos A \cos a}{1 - \sin a \sin b \sin A \sin B}, \\
\cos C &= \frac{-\cos A \cos B + \sin A \sin B \cos a \cos b}{1 - \sin a \sin b \sin A \sin B}, \\
c &= \text{atan2}(\sin c,\, \cos c).
\end{aligned}$$

This function is used only internally. Instructions on how to use it manually are given in the help.

`acos2.m`, `acos2d.m`, `H.m` and `Hd.m`

`H.m` and `Hd.m` are the implementations of the Hemisphere-function $H(x)$, which determines the sign of the arccosine in `acos2.m` or `acos2d.m`.

This function is used only internally. Instructions on how to use it manually are given in the help.

`blkassign.m`

This is a tool that prevents long blocks of assignments, when each variable to be assigned is a column or row of a given matrix. Normally, this is done as

$$\begin{aligned}
A &= \texttt{rand}(5); \\
a &= A(:,\, 1); \\
b &= A(:,\, 2); \\
c &= A(:,\, 3); \\
 & \quad \text{etc.}
\end{aligned}$$

but with `blkassign.m`, this can be shortened as

$$\begin{aligned} A &= \texttt{rand}(5); \\ [a,\, b,\, c,\, \ldots] &= \texttt{blkassign}(A); \end{aligned}$$

Also this function is used internally only. Instructions on how to use it manually are given in the help.

Bibliography

[Wertz(2001)] James R. Wertz. *Mission Geometry: Orbit and Constellation Design and Management.* Published Jointly by Microcosm Press and Kluwer Academic Publishers, 2001.