A MATLAB Script for Predicting Equinoxes and Solstices

This document describes a MATLAB script named `equsol.m` which determines the time of the equinoxes and solstices of the Earth. These events are the times when the apparent geocentric longitude of the Sun is an exact multiple of 90 degrees. This script uses Brent's root-finder and a precision solar ephemeris to calculate these events.

Brent's method requires an objective function that defines the nonlinear equation to be solved. The objective function for the spring and fall equinoxes is the geocentric declination of the Sun. The spring and fall equinoxes occur whenever the geocentric declination of the Sun is less than or equal to a user-defined convergence criterion.

For the summer and winter solstices, the objective function is $\theta - \lambda_s$, where $\theta = 90^\circ$ for the summer solstice, $\theta = 270^\circ$ for the winter solstice, and λ_s is the geocentric longitude of the Sun. The summer and winter solstices occur whenever the difference $\Delta = \theta - \lambda_s$ is less than or equal to a user-defined convergence criterion.

Brent's method also requires an initial and final time which bounds the root of the objective function. The initial time for the spring equinox is March 15, for the summer solstice June 15, for the fall equinox September 15 and for the winter solstice December 15. For each event, the final time is equal to these initial dates plus 10 days.

The `equsol` script will prompt you for the calendar year of interest. The following is a typical user interaction with this MATLAB script. Please note that each event is displayed in UTC time.

```
time of the equinoxes and solstices
===================================

please input the calendar year
? 2013

spring equinox
--------------

calendar date         20-Mar-2013
UTC time              11:01:41.257

summer solstice
---------------

calendar date         21-Jun-2013
UTC time              05:03:50.335

fall equinox
------------

calendar date         22-Sep-2013
UTC time              20:43:51.898

winter solstice
---------------

calendar date         21-Dec-2013
UTC time              17:10:59.660
```

The following are the results for this same calendar year using the Multiyear Interactive Computer Almanac (MICA) published by the United States Naval Observatory.

SOLSTICES AND EQUINOXES

	Equinox	Solstice	Equinox	Solstice
Year	Date Time (UT) d h m	Date Time (UT) d h m	Date Time (UT) d h m	Date Time (UT) d h m
2013	Mar 20 11:02	Jun 21 5:04	Sep 22 20:44	Dec 21 17:11

Time conversion

The fundamental time argument for the solar ephemeris function used in this MATLAB script is "ephemeris" time. As implemented here, we assume this time argument to be Barycentric Dynamic Time (TDB). To report the time of these celestial events in Universal Coordinated Time (UTC) or civil time, we need an algorithm to make this time conversion.

The following is the MATLAB source code for a function which iteratively performs this calculation using Brent's root-finding method.

```
function jdutc = tdb2utc (jdtdb)

% convert TDB julian date to UTC julian date

% input

%  jdtdb = TDB julian date

% output

%  jdutc = UTC julian date

% Orbital Mechanics with MATLAB

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global jdsaved

jdsaved = jdtdb;

% convergence tolerance

rtol = 1.0d-8;

% set lower and upper bounds

x1 = jdsaved - 0.1;

x2 = jdsaved + 0.1;

% solve for UTC julian date using Brent's method

[xroot, froot] = brent ('jdfunc', x1, x2, rtol);
```

```
jdutc = xroot;

end
```

This function calls the following MATLAB objective function during the calculations.

```
function fx = jdfunc (jdin)

% objective function for tdb2utc

% input

%  jdin = current value for UTC julian date

% output

%  fx = delta julian date

% Orbital Mechanics with MATLAB

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global jdsaved

tai_utc = findleap(jdin);

fx = utc2tdb (jdin, tai_utc) - jdsaved;

end
```

Notice that this function requires the `findleap` function which calculates the number of leap seconds for the current UTC Julian date value. The `jdfunc` function is computing the difference between the TDB Julian date input by the user and the value computed by the `utc2tdb` MATLAB function. The algorithm has converged whenever this value is less than or equal to the user-defined tolerance `rtol`.