**1 概述**

最近在学习GPS解算算法时需要在GPS时（GPS周和周内秒）和公历日期之间进行转换，于是就整理了一些时间转换的小程序。本文介绍了GPS时、公历、儒略日（JD）、简化儒略日（MJD）之间的转换函数。

|  |  |
| --- | --- |
| gps2cal | 将GPS周和周内秒转换到公历时间 |
| gps2cal1 | 由公历日期和GPS周内秒计算公历时间 |
| cal2gps | 将公历GPS时间转换到GPS周和周内秒 |
| mjd2cal | 将简化儒略日转换到公历时间 |
| cal2mjd | 将公历时间转换到简化儒略日 |
| jd2cal | 从儒略日计算公历时间 |
| cal2jd | 将公历时间转换到儒略日 |
| cal2wd | 借助MJD，由公历年月日推算星期几，按照格里高利十三世的历法改革去掉1582年10月5日至14日 |
| cal2wd1 | 由公历年月日推算星期几，按照英国人的做法去掉1752年9月3日至13日 |
| cal2wd2 | 不借助MJD，由公历年月日推算星期几，去掉1582年10月5日至14日 |

在MJD基础上推算星期是很简单的，我顺带写出了从公历推算星期的函数cal2wd。不借助MJD也可以推算星期，例如安竹林同学在《再谈星期的计算》（《程序员》2001年第4期）介绍的方法。本文给出了这个方法的matlab版本cal2wd1。 cal2wd1按照英国人的历法去掉了公历中的公元1752年9月3日至13日。我又写了cal2wd2函数，这个函数按照格里高利十三世的做法去掉1582年10月5日至14日。

我在2005年写过一篇[《时标与历法》](http://www.fmddlmyy.cn/text8.html)，介绍过一些历法的基础知识。本文假设读者了解该文介绍的一些概念，例如原子时标、UTC、闰秒、儒略历、儒略日、简化儒略日等。

**2 GPS周和周内秒**

GPS时从1980年1月6日0时开始计时。GPS时的秒长度与UTC一样，采用原子时标。但GPS时是连续的，不调整闰秒。在GPS数据中，GPS时通常被表示为GPS周和周内秒。例如：

>> gpst=cal2gps([2009 3 13 22 38 10])

gpst =

1522 513490

>> cal=gps2cal(gpst)

cal =

2009 3 13 22 38 10

即2009年3月13日22:38:10对应gps周1522，周内秒513490。 GPS数据为什么要采用周和周内秒的表示方式呢？

因为这样便于存储和处理。GPS系统内部每隔1.5s会产生一个叫作X1历元的定时事件。系统通过对X1历元计数的方式计时，这种计时方式被称作Z计数。 Z计数有29bit，高10bit是周数，低19bit是周内秒。一个星期有604800秒，19bit所能表示的最长时间是(2^19-1)\*1.5=786430.5秒，够用啦。 GPS卫星和接收机之间通过Z计数传递时间，所以GPS数据文件中会用到GPS周和周内秒。

我在做数据解算时，有时知道观测日期和周内秒，所以写了一个由公历日期和GPS周内秒计算公历时间的函数gps2cal1：

>> cal=gps2cal1([2009 3 13],513490)

cal =

2009 3 13 22 38 10

请注意这3个函数里的公历时间没有调整闰秒。在GPS数据（例如RINEX格式）中，公历时间也不调整闰秒。

**3 儒略日和简化儒略日**

我在[《时标与历法》](http://www.fmddlmyy.cn/text8.html)中介绍过儒略历、儒略日（JD）和简化儒略日（MJD），这里就不再重复。只是简单说明一下公历的不连续性。公元1582年3月1日，罗马教皇格里高利十三世在颁布格里高利历（即我们现在使用的公历）时，为了消除已经存在的误差，宣布去掉1582年10月5日至14日。即1582年10月4日的23:59的下一秒是1582年10月15日的00:00。

我们通常对1582年10月15日0时之后的时间使用格里高利历，对1582年10月4日的23:59之前的时间使用儒略历。这里的儒略历是指屋大维改革过的儒略历，即奥古斯都历。它和格里高利历除了置闰方式不同，其它是完全一样的。儒略历是4年一闰，400年100闰。公历是400年97闰，扣掉了3个不能被400整除的世纪年。

例如：香港天文台说儒略日“以倒推到公元前4713年1月1日格林尼治平时正午为起算日期”。这里的公元前4713年1月1日就是按照儒略历倒推的日期。但没有必要特殊说明儒略历。本文提供的mjd2cal、cal2mjd、jd2cal、cal2jd函数都使用了规则：公元1582年10月4日24:00点之前使用儒略历，公元1582年10月15日00:00点之后使用格里高利历。下面是一些使用示例：

>> jd2cal(0)

ans =

-4713 1 1 12 0 0

>> mjd2cal(0)

ans =

1858 11 17 0 0 0

>> cal2jd([1582 10 4 12 0 0])

ans =

2299160

>> cal2jd([1582 10 15 12 0 0])

ans =

2299161

>> cal2mjd([1582 10 4 0 0 0])

ans =

-100841

>> cal2mjd([1582 10 15 0 0 0])

ans =

-100840

网上有很多介绍闰年的文章会以公元1000年为例，说它不是闰年。事实上，按照1582年10月4日24:00点之前使用儒略历的规则，公元1000年是闰年。也就是说公元1000年2月29日在历史上是存在的。例如：

>> cal2jd([1000 2 28 12 0 0])

ans =

2086366

>> cal2jd([1000 2 29 12 0 0])

ans =

2086367

>> cal2jd([1000 3 1 12 0 0])

ans =

2086368

**3 星期推算**

在MJD的基础上，我们可以很方便地推算星期。函数cal2wd只需要两行代码：

mjd=floor(cal2mjd(cal));

% 2009年3月9日（MJD 54899）是星期一

wd=mod(mjd-54899,7)+1;

因为在JD或MJD中，1582年10月5日至14日是不存在的。所以1582年10月15日是星期五，1582年10月4日是星期四：

>> cal2wd([1582 10 15])

ans =

5

>> cal2wd([1582 10 4])

ans =

4

不借助MJD，也可以直接推算星期。安竹林同学在《再谈星期的计算》一文中介绍了一个简单易行的方法。我将其写成matlab函数cal2wd1。不过在安同学的代码中，1582年10月5日至14日是存在的，1752年9月3日至13日是不存在的，这是怎么回事呢？

事情是这样的。1582年3月1日格里高利十三世颁布格里高利历后，英国人认为从历史上去掉10天是很荒唐的事情，没有接受。其实更合理的解释是：格里高利历是罗马教皇颁布的，信奉天主教的意大利、波兰、西班牙、葡萄牙都国家都很痛快地接受了。但英国是新教国家，所以不愿意接受。直到1752年，英国人才决定采用格里高利历，因为这时儒略历又多算了一天，所以英国人从历史上去掉了11天。在英国人看来，1752年9月2日的次日是1752年9月14日。

在我看来，还是去掉1582年10月5日至14日更常用一些。所以又在cal2wd1基础上写了cal2wd2。 cal2wd2和cal2wd功能相同，区别仅是前者不依赖cal2mjd函数。cal2wd和cal2wd1也就是对1582年10月5日至1752年9月13日的推算结果不同。它们对其它日期的推算结果都是相同的，例如它们推算的公元1年1月1日都是星期六：

>> cal2wd([1752 9 14])

ans =

4

>> cal2wd1([1752 9 14])

ans =

4

>> cal2wd([1 1 1])

ans =

6

>> cal2wd1([1 1 1])

ans =

6

**4 结束语**

本文提到的源代码可以从[这里](http://www.fmddlmyy.cn/down2/calm.rar)下载。其中最复杂的cal2jd和jd2cal是根据[www.moshier.net](http://www.moshier.net/" \t "_top)上的代码修改得到。其实我起初只是想写两个转换gps时间的函数。好奇心让我在半夜查阅相关资料，并用本文记录了这些函数的一些背景。

时间转换--年积日-儒略日-GPS周

一、年积日、儒略日、GPS周等简介

儒略历

公元前46年，罗马执政官儒略·凯撒颁布了儒略历。儒略历每年12个月，平年365天，闰年366天。除2月外，单数月份31天，偶数月份30天。2月份平年29天，闰年30天。每隔3年置一闰年。

儒略历是纯太阳历，每年的平均长度是365.25天。每400年，儒略历与回归年大约相差3天，即：(365.25-365.242189)\*400=3.1244凯撒为了纪念改历成功，将他出生的7月从Quintilis改成自己的名字Julius

儒略日和简化儒略日

在天文学有一种连续纪日的儒略日（JD），它以儒略历公元前4713年1月1日的GMT正午为第0日的开始。还有一种简化儒略日（MJD）：MJD=JD-2400000.5

MJD的第0日是从公历1858年11月17日的GMT零时开始的。我写完前一个句号时的MJD是53583.22260。小数部分是以UTC时间在当天逝去的秒数除以86400得到的。0.22260约为UTC时间的5:20，加上中国的时区就是13:20。

年积日

年积日是仅在一年中使用的连续计时法。每年的1月1日计为第1日，2月1日为第32日。平年的12月31日为第365日，闰年的12月31日为第366日。

格里高利历——公历

公元1582年3月1日，罗马教皇格里高利十三世颁布了格里高利历，将不能被4整除的世纪年算作平年，这就是我们现在使用的公历。

在公历中，每400年有97个闰年，平均每年的长度是：

(365\*400+97)/400=365.2425天

每400年，公历与回归年大约相差0.1244天，即：

(365.2425-365.242189)\*400=0.1244

GPS时、GPS周、周内秒

GPS周（GPSWeek）是GPS系统内部所采用的时间系统。

时间零点定义的为：1980年1月5日夜晚与1980年1月6日凌晨之间0点。最大时间单位是周（一周：604800秒）。每1024周（即7168天）为一循环周期。第一个GPS周循环点为1999年8月22日0时0分0秒。即从这一刻起，周数重新从0开始算起。星期记数规则是：Sunday为0，Monday为1，以此类推，依次记作0~6，GPS周记数（GPSWeekNumber）为“GPS周星期记数”。

表示方法：从1980年1月6日0时开始起算的周数加上周内时间的秒数（从每周周六/周日之夜开始起算的秒数，例如：1980年1月6日0时0分0秒的GPS周：第0周，第0秒。

例如：2004年5月1日10时5分15秒的GPS周：第1268周，

GPS周记数第554715秒，（GPSWeekNumber）为12686，第554715

秒。

三、程序实现成果(具体代码见附件)

1.gps2cal将GPS周和周内秒转换到公历时间

functioncal=gps2cal(gpst)

%gps2cal将GPS周和周内的秒转换到公历GPS时间

%cal=gps2cal(week,sec)返回的公历是1x6矩阵，6列分别为年月日时分秒%gpst：1x2矩阵，2列分别为GPS周和周内的秒

%GPS从MJD44244开始

mjd=44244+(gpst(1)\*86400\*7+gpst(2))/86400;

cal=mjd2cal(mjd);

2.gps2cal1由公历日期和GPS周内秒计算公历时间functioncal=gps2cal1(date,tow)

%gps2cal1由公历日期和gps周内秒计算公历GPS时间

%cal=gps2cal1(date,tow)返回的公历是1x6矩阵，6列分别为年月日时分秒%date：1x3矩阵，3列分别为公历年月日

%tow：GPS周内秒

mjd=cal2mjd(date);

%GPS从MJD44244开始

week=floor((mjd-44244)/7);

cal=gps2cal([week,tow]);

3.cal2gps将公历GPS时间转换到GPS周和周内秒functiongpst=cal2gps(cal)

%cal2gps将公历GPS时间转换到GPS周和周内的秒。

%gpst==cal2gps(cal)返回的gpst是1x2矩阵，2列分别为GPS周和周内秒%cal：1x6矩阵，6列分别为年月日时分秒。构造cal时可以省略末尾的0iflength(cal)<6

cal(6)=0;

end

mjd=cal2mjd(cal);

%GPS从MJD44244开始

elapse=mjd-44244;

week=floor(elapse/7);

elapse=elapse-week\*7;%周内天数

gpst=[weekround(elapse\*86400)];

4.mjd2cal将简化儒略日转换到公历时间

functioncal=mjd2cal(mjd)

%mjd2cal将简化儒略日转换到公历年月日时分秒。

%cal=mjd2cal(mjd)返回的cal是1x6矩阵，6列分别为年月日时分秒%mjd：简化儒略日

jd=mjd+2400000.5;

cal=jd2cal(jd);

5.cal2mjd将公历时间转换到简化儒略日

functionmjd=cal2mjd(theyear,themonth,theday);

%returnthejulianday(year,day)定义函数julia为转换为julian日期函数，变量y为返回的julian日期，输入变量为theyear/年，themonth/

月，theday/日

leapyear=rem(theyear,4);%年对4求余数

ifleapyear>0

leap=0;%判断余数大于0，不能被4整除，则用leap标记这一年不是

闰年（标记为0）

else

ifrem(theyear,100)==0&rem(theyear,400)~=0

leap=0;%能被4整除但是不能被400整除也不是闰年，则用leap标记

这一年不是闰年

leap=1;

end

end

%%%%%%采用平年的日历，按1-12月分月计算julian日期%其他情况是闰年

ifthemonth==1

juliaday=theday;

end

ifthemonth==2

juliaday=theday+31;

end

ifthemonth==3

juliaday=theday+59;

end

ifthemonth==4

juliaday=theday+90;

end

ifthemonth==5

juliaday=theday+120;

end

ifthemonth==6

juliaday=theday+151;

end

ifthemonth==7

juliaday=theday+181;

end

ifthemonth==8

juliaday=theday+212;

end

ifthemonth==9

juliaday=theday+243;

end

ifthemonth==10

juliaday=theday+273;

end

ifthemonth==11

juliaday=theday+304;

end

ifthemonth==12

juliaday=theday+334;

end

ifleap==1

ifthemonth<=2

juliaday=juliaday;%如果是闰年，当月份小于2月时，julian日期与平年相同

ifthemonth>=3

juliaday=juliaday+1;

end

end

mjd(1)=theyear;

mjd(2)=juliaday;

日%当月份大于2月时，在平年的基础上加一天%返回的矢量y，第一个元素为年，第二个元素为Julian

6.jd2cal从儒略日计算公历时间

functioncal=jd2cal(J)

%从儒略日计算公历年月日时分秒

%cal=jd2cal(J)

%返回的cal是1x6矩阵，6列分别为年月日时分秒

%

%公元1582年10月4日24:00点之前使用儒略历，公元1582年10月15日00:00点之后使用公历

if(J<1721423.5)

BC=1;

else

BC=0;

end%公元1月1日0时

%startfromJulianMarch1,4801B.C.

if(J<2299160.5)%before1582.10.4.24:00isJuliancalender

j0=floor(J+0.5);

dd=J+0.5-j0;

else%after1582.10.15.00:00isGregoriancalender