**Negative Jump Algorithm**

When calculating the ticks associated with a UTC, add a before or after jump flag if the UTC value is within the ambiguous range. A value of 0 indicates before the jump and 1 indicates after the jump. We need to add the flag to the createTicks function (and make sure the createTicks function is always used – there are times when the tick struct is loaded directly). The flag will be unnecessary, except for the 0.1 second and 0.2 second ambiguous intervals.

Will there ever be a situation where we have a tick count that was not created from a UTC, so don’t know the value of the before or after flag. This can occur if we allow the addition or subtraction of a relative tick count to a UTC. For the time being, we want to allow this. In the future, when we have a relative CalCoords type, we can restrict the addition or subtraction to these types. We could still create a relative tick by subtracting two UTCs, but not allow a relative tick to be added to a UTC. For the time being, we could adopt the following approach:

When a relative tick count is added to or subtracted from a UTC and the resulting tick count is in the ambiguous region, do the following:

If the UTC is before the start of the ambiguous region, set the flag to before

If the UTC is after the end of the ambiguous region, set the flag to after

If the UTC is within the ambiguous region, preserve the value of the flag

When subtracting raw tick counts, not as a result of a UTC subtraction, ignore the before or after flag. This will produce the intuitive result.

When subtracting tick counts in support of a UTC subtraction, we need to respect the before or after flags to produce an intuitive result.

If neither is within the ambiguous region, there is no issue.

If both values are within the ambiguous range:

If both values have the same before or after flag, just subtract

If the flags are different, add .05 to the absolute value of the difference

**Improved Algorithm**

Taking a step back, a positive jump indicates the UTC clock was ticking too slowly during the previous period. This means each UTC second should have been shorter. Thus, a “correct” computation of the difference between 2 UTC times (in terms of TAI seconds) during the previous period would have been slightly shorter. However, the formalism (i.e., the accepted standard) is to use the specified UTC second length for that period.

At the end of the period, a jump is applied. Thus, an attosecond step across the jump in UTC time is 0.1 (if that is the jump length), plus an attosecond in TAI time. The question is should we record the difference between these two UTC times as 1 attosecond or 0.1 + 1 attosecond. The standard seems clear, that the result should be 0.1 + 1 attosecond. This is equivalent to making up the entire shortfall accumulated during the previous period in 1 attosecond.

Similarly, a negative jump indicates the UTC clock was ticking too quickly during the previous period. This means each UTC second should have been longer. The standard for that period is to use the specified UTC second length.

So far, this is consistent with the positive jump methodology. However, if we were to continue with that approach across the jump, we would have a very non-intuitive result; namely, the 1 attosecond step across the jump would be recorded as 1 attosecond – 0.1 second. This would suggest a forward step in UTC time is a backward step in TAI time. In the positive jump case, we were saying a forward step in UTC time is a longer forward step in TAI time, which is more intuitive.

An intuitive solution is to add the negative jump length when computing the difference between two UTC times that straddle the jump. The only problem with this approach is that the accumulated overage during the period would never be recognized – i.e., a UTC time difference across the jump measured in TAI seconds would always be recorded as 0.1 seconds too long.

A good compromise is to apply adjustment to the result of the computed difference between two UTC values over only a limited range of values. Two reasonable options are to apply an adjustment only when the difference would have the wrong sign, or only when both values are within the ambiguous interval.

There are two times when negative jumps occur, 8/1/1961 00:00:00 UTC and 2/1/1968 00:00:00 UTC. Consider the first time. Prior to the jump, TAI – UTC is 1.69757 seconds. After the jump, TAI – UTC is 1.64757 seconds. This means that, for example, 8/1/1961 00:00:01.64757 corresponds to both 7/31/1961 23:59:59.95 UTC and 8/1/1961 00:00:00 UTC. Negative differences will be computed when 8/1/1961 00:00:00 <= UTC1 < 8/1/1961 00:00:00.05 and 7/31/1961 23:59:59.95 < UTC2 < 8/1/1961 00:00:00, whenever UTC2 > UTC1 - .05.

For example, if UTC1 = 8/1/1961 00:00:00.01 and UTC2 = 7/31/1961 23:59:59.98, then TAI1 = 8/1/1961 00:00:01.65757 and TAI2 = 8/1/1961 00:00:01.67757. The difference would be -.02 without adjustment and the more intuitive .03 with adjustment. If we instead consider UTC1 = 8/1/1961 00:00:00.04, then TAI1 = 8/1/1961 00:00:01.68757 and the difference would be .01, instead of the more intuitive .06.

The second approach eliminates the whenever clause. That is, an adjustment is made whenever 00:00:00 <= UTC1 < 8/1/1961 00:00:00.05 and 7/31/1961 23:59:59.95 < UTC2 < 8/1/1961 00:00:00. The second example in the previous paragraph would have the more intuitive result of .06. If we consider the case where UTC1 = 8/1/1961 00:00:00.049 and UTC2 = 7/31/1961 23:59:59.99, the adjusted difference would be .05. However, if we increase UTC1 to 8/1/1961 00:00:00.05, there would be no adjustment, so the difference would be .001, instead of .051.

A third approach, which is likely the most intuitive, is to apply an adjustment if either UTC1 or UTC2 is within the ambiguous range and the other value is on the other side of the jump, whether within the ambiguous range or not. This has the advantage of being easier to explain and understand for cases when UTC1 > UTC2 and UTC1 < UTC2, and still limits cases where the deviations from the standard to only cases where at least 1 of the values is within the ambiguous range, while limiting non-intuitive changes in results (as in the last example of the previous paragraph) to situations where we move from 1 value being in the ambiguous range to no values being within the ambiguous range. This has the effect of recognizing the accumulated effect of the overage in 0.1 second (i.e., between 7/31/1961 23:59:59.95 and 8/1/1961 00:00:00.05, while avoiding sign errors in the difference calculation.

Below are a few examples with the recommended approach. The ambiguous range is 7/31/1961 23:59:59.95 <= UTC < 8/1/1961 00:00:00.05. The range is closed at the bottom and open at the top, because 8/1/1961 00:00:00 UTC has the same TAI value as 7/31/1961 23:59:59.95 UTC.

1. UTC1 = 8/1/1961 00:00:00.06, UTC2 = 7/31/1961 23:59:59.94. Neither value is within the ambiguous range, so UTC1 – UTC2 is 0.07, instead of 0.12
2. UTC1 = 8/1/1961 00:00:00.04, UTC2 = 7/31/1961 23:59:59.96. Both values are within the ambiguous range, so UTC1 – UTC2 is 0.08.
3. UTC1 = 8/1/1961 00:00:00.05, UTC2 = 7/31/1961 23:59:59.95. UTC2 is within the ambiguous range on the opposite side of the jump, so UTC1 – UTC2 is 0.10.
4. UTC1 = 8/1/1961 00:00:00.05, UTC2 = 7/31/1961 23:59:59.949. Neither value is within the ambiguous range, so UTC1 – UTC2 is 0.051, instead of 0.101.
5. UTC1 = 8/1/1961 00:00:00.1, UTC2 = 8/1/1961 00:00:00. UTC2 is within the ambiguous range, but on the same side of the jump, so UTC1 – UTC2 = 0.1.
6. UTC1 = 8/1/1961 00:00:00.1, UTC2 = 7/31/1961 23:59:59.99. UTC2 is within the ambiguous range, on the other side of the jump, so UTC1 – UTC2 is 0.11.

Special handling for negative jumps is only required for a limited set of operations involving UTC’s. The jump can be handled as part of the algorithm for those operations, as follows:

\*\*\* what about UTC1 and UTC2 in different ambiguous ranges? No adjustment required on subtraction. \*\*\*

\*\*\*\*\*\* Change the following to be consistent with the third approach above. \*\*\*\*\*

1. UTC1 – UTC2 = RTAI
   1. UTC1 > UTC2 if UTC1 > UTC transition and UTC2 < UTC transition, add .05. Otherwise, no adjustment.
      1. UTC1 = 8/1/1961 00:00:00.02 -> TAI1 = 8/1/1961 00:00:01.66757
      2. UTC2 = 7/31/1961 23:59:59.96 -> TAI2 = 8/1/1961 00:00:01.65757
      3. TAI1 – TAI2 = .01, but we’d prefer a result of .06
   2. UTC1 < UTC2 if UTC1 < UTC transition and UTC2 > UTC transition, subtract .05. Otherwise, no adjustment.
      1. UTC1 = 7/31/1961 23:59:59.90 -> TAI1 = 8/1/1961 00:00:01.59757
      2. UTC2 = 8/1/1961 00:00:00.08 -> TAI2 = 8/1/1961 00:00:01.72757
      3. TAI1 – TAI2 = -.13, but we’d prefer a result of -.18
2. UTC1 + RTAI = UTC2
   1. UTC1 < UTC transition, RTAI > UTC transition – UTC1. Add .05. Should it be >=?
3. UTC1 – RTAI = UTC2

\*\*\* Can we say there is only a potential issue if TAI1 or TAI1 +/- RTAI is within an ambiguous TAI range? \*\*\*

If UTC1 is within the ambiguous range, first determine which side of the jump. If on the positive side and RTAI results in an increase (either plus a positive value or minus a negative value), no adjustment necessary.

If UTC1 is not within the ambiguous range,

Algorithm for Computing UTC1 + RTAI Considering Ambiguous Regions

UTC1 before jump?

Start

UTC1 within ambiguous region?

RTAI > 0?

TAI1 + RTAI within ambiguous region?

TAI1 + RTAI >= TAIjump?

RTAI > 0?