Celestial to Terrestrial Transformation

A CIO Based Matrix Algebra Approach using MATLAB

Abel Brown

November 20, 2012

Chapter 1

Definitions

The transformation to be used to relate the International Terrestrial Reference System (ITRS) to the Geocentric Celestial Reference System (GCRS) at the date t of the observation can be written as:

$$[GCRS] = Q(t)R(t)W(t)[ITRS] \tag{1.1}$$

where $Q(t)$, $R(t)$, and $W(t)$ are the transformation matrices arising from the motion of the celestial pole in the celestial reference system, from the rotation of the Earth around the axis associated with the pole, and from polar motion respectively. Note that Eq. (1.1) is valid for any choice of celestial pole and origin on the equator of that pole. The definition of the GCRS and ITRS and the procedures for the ITRS to GCRS transformation that are provided in this document comply with the IAU 2000/2006 resolutions. Numerical values contained in this document are compliant with the IAU 2006 precession. More detailed explanations about the relevant concepts, software, and IERS products corresponding to the IAU 2000 resolutions can be found in IERS Technical Notes 29, 32, and 36

The *Celestial Intermediate Origin* (CIO) based method is the newest technique, but is also most different from the previous equinox and IAU76/FK5 implementations. This CIO based method effectively eliminates the distinctions of the 'equinox' and 'mean position of date'. The process rests with the determination of the CIO coordinates; X, Y, and s. These are the *Celestial Intermediate Pole* (CIP) coordinates in the GCRS. Sidereal time is replaced by *Earth Rotation Angle* (ERA). The Greenwich meridian is replaced by an International Terrestrial Reference Frame (ITRF), and a Terrestrial Intermediate Origin (TIO) is used for the rotation frame. These, in turn, form a transformation matrix that is used to convert vectors between systems. There are no "mean" positions - only the *International Celestial Reference Frame* (ICRF) and true positions - in this system. See figure (1.1) for more details.

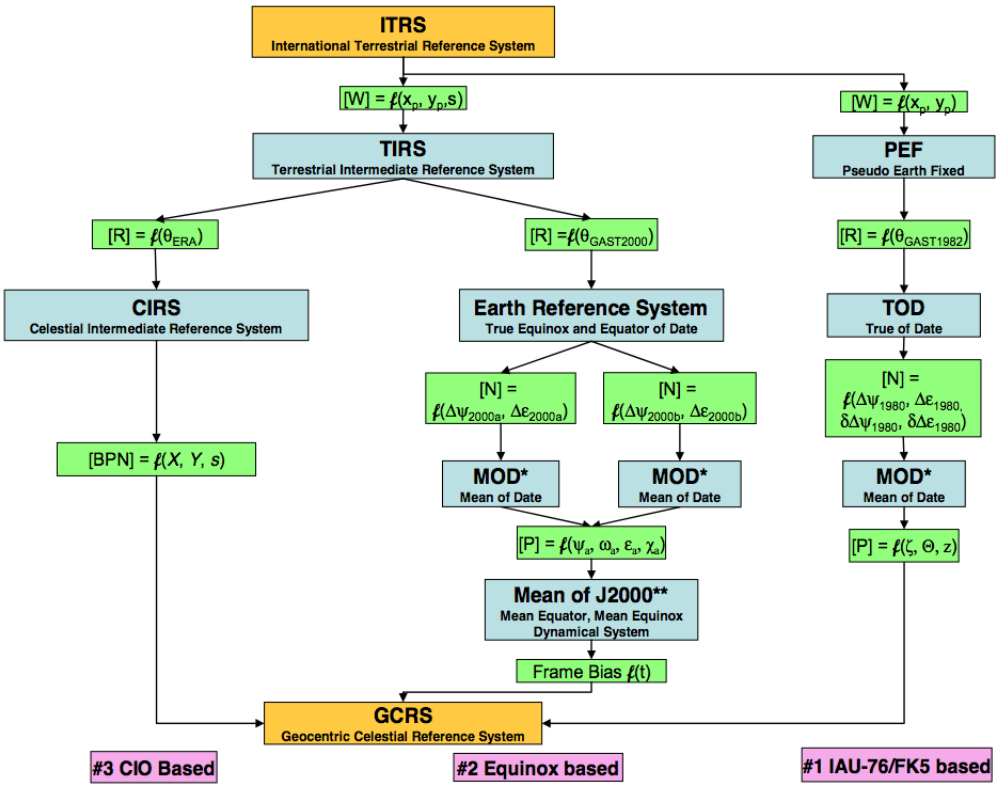

Figure 1.1: *Celestial to Terrestrial Coordinate Transformations.* Three general approaches (and two variations) transform vectors between terrestrial (ITRS) and celestial coordinate systems (GCRS). Although names (e.g polar motion, mean-of-date, etc) are the same, the formulae are different. IAU-76/FK5 precession/nutation corrections are found in the EOP data. There are also two approaches for the equinox-based precession/nutation models, IAU-2000A and IAU 2000B

Expressions for the transformation matrix for polar motion

The transformation matrix arising from polar motion (i.e. relating TIRS and ITRS) can be expressed as:

$$W_{TIRS\to ITRS}(t) = R_1(-y_p) \cdot R_2(-x_p) \cdot R_3(s') \tag{1.2}$$

where x_p and y_p are the polar coordinates of the CIP in the ITRS and s' being a quantity, named "TIO locator", which provides the position of the *Terrestrial Intermediate Origin* (TIO) on the equator of the CIP corresponding to the kinematical definition of the "non-rotating" origin (NRO) in the ITRS when the CIP is moving with respect to the ITRS due to polar motion. The use of the quantity s', which was neglected in the classical form prior to 1 January 2003, is necessary to provide an exact realization of the "instantaneous prime meridian" (designated by "TIO meridian").

The quantity s' (i.e. the TIO locator) appearing in Eq. (1.2) is sensitive only to the largest variations in polar motion. Some components of s' have to be evaluated, in principle, from the measurements and can be extrapolated using the IERS data. Its main component can be written as $s' = -0.0015(a_c^2/1.2 + a_a^2)t$ where a_c and a_a are the average amplitudes (in arc-seconds) of the Chandlerian and annual wobbles, respectively, in the period considered (Capitaine et al., 1986). The value of s' will therefore be less than 0.4 mas through the next century, even if the amplitudes for the Chandlerian and annual wobbles reach values of the order of $0.5''$ and $0.1''$, respectively. Using the current mean amplitudes for the Chandlerian and annual wobbles (Lambert and Bizouard, 2002) gives s' in micro arc-seconds (μas) as:

$$s' = -47 \cdot t_c \tag{1.3}$$

where t_c is in Julian centuries since J2000 referenced to Terrestrial Time (TT) [1]

Expression for the CIO based transformation matrix for Earth rotation

The CIO based transformation matrix arising from the rotation of the Earth around the axis of the CIP (i.e. relating CIRS and TIRS), can be expressed as:

$$R_{CIRS\to TIRS}(t) = R_3(ERA) \tag{1.4}$$

where ERA is the Earth Rotation Angle between the CIO and the TIO at date t on the equator of the CIP, which provides a rigorous definition of the sidereal rotation of the Earth.

The conventional relationship defining UT1 in radians from the Earth Rotation Angle (ERA) to be used is that given by Capitaine et al. (2000):

$$ERA(T_{UT1}) = 2\pi(F_{UT1} + 0.7790572732640 + 0.00273781191135448 \cdot T_{UT1}) \tag{1.5}$$

[1] That is, $t_c = (2400000.5 - 2451545 + \text{fMJD}_{TT})/36525$.

where F_{UT1} is the UT1 Julian day fraction and T_{UT1} =Julian UT1 date - 2451545.5. Note that ERA should be normalized between 0 and 2π. This definition of UT1 based on the CIO is insensitive at the micro-arcsecond level to the precession-nutation model and to the observed celestial pole offsets. Note also that, for 0h UT1, the UT1 Julian day fraction in Eq. (1.5) is 0.5. Similarly to polar motion, additional components should be added to the values published by the IERS for UT1 and LOD to account for the effects of ocean tides and libration[2].

Expression for the transformation matrix for the celestial motion of the CIP

The CIO based transformation matrix arising from the motion of the CIP in the GCRS (i.e. relating GCRS and CIRS), can be expressed as:

$$Q_{GCRS \to CIRS}(t) = R_3(-(E+s)) \cdot R_2(d) \cdot R_3(E); \tag{1.6}$$

E and d being such that the coordinates of the CIP in the GCRS are:

$$X = \sin(d)\cos(E), \qquad Y = \sin(d)\sin(E), \qquad Z = \cos(d) \tag{1.7}$$

and s being a quantity, named "CIO locator", which provides the position of the CIO on the equator of the CIP corresponding to the kinematical definition of the NRO in the GCRS when the CIP is moving with respect to the GCRS, between the reference epoch and the date t due to precession and nutation. Thus, provided X and Y in radians, the quantities E and d are:

$$E = atan2(Y, X), \qquad d = atan\left(\sqrt{\frac{X^2+Y^2}{1-(X^2+Y^2)}}\right) \tag{1.8}$$

Expression for X, Y, and s

The coordinates of the CIP in the GCRS to be used for the parameters X and Y appearing in Eq. (1.8) can be given by developments as function of time of those quantities. Developments valid at the microarcsecond level, based on the IAU 2006 precession and IAU 2000A nutation and on their corresponding pole and equinox offsets at J2000.0 with respect to the pole of the GCRS have been computed (Capitaine and Wallace, 2006). They replace the previous developments based on the IAU 2000 model for precession-nutation and frame biases that had been provided by Capitaine et al. (2003a) and in the IERS 2003 Conventions.

The IAU 2006/2000A developments for X, Y, and s in microarcseconds, are as follows:

[2]See UTLIBR.F provided by the IERS at http://tai.bipm.org/iers/

$$
\begin{aligned}
X = &- 16617t^0 + 2004191898t^1 - 429782.9t^2 - 198618.34t^3 + 7.578t^4 + 5.9285t^5 \\
&+ \sum_{i}^{1306} [\, s_{x,0,i}\ \sin(ARG) + c_{x,0,i}\ \cos(ARG)\,] \cdot t^0 \\
&+ \sum_{j}^{253} [\, s_{x,1,j}\ \sin(ARG) + c_{x,1,j}\ \cos(ARG)\,] \cdot t^1 \\
&+ \sum_{k}^{36} [\, s_{x,2,k}\ \sin(ARG) + c_{x,2,k}\ \cos(ARG)\,] \cdot t^2 \\
&+ \sum_{l}^{4} [\, s_{x,3,l}\ \sin(ARG) + c_{x,3,l}\ \cos(ARG)\,] \cdot t^3 \\
&+ \sum_{m}^{1} [\, s_{x,4,m}\ \sin(ARG) + c_{x,4,m}\ \cos(ARG)\,] \cdot t^4
\end{aligned}
\tag{1.9}
$$

$$
\begin{aligned}
Y = &- 6951t^0 - 25896t^1 - 22407274.7t^2 + 1900.59t^3 + 1112.526t^4 + 0.1358t^5 \\
&+ \sum_{i}^{962} [\, s_{y,0,i}\ \sin(ARG) + c_{y,0,i}\ \cos(ARG)\,] \cdot t^0 \\
&+ \sum_{j}^{277} [\, s_{y,1,j}\ \sin(ARG) + c_{y,1,j}\ \cos(ARG)\,] \cdot t^1 \\
&+ \sum_{k}^{30} [\, s_{y,2,k}\ \sin(ARG) + c_{y,2,k}\ \cos(ARG)\,] \cdot t^2 \\
&+ \sum_{l}^{5} [\, s_{y,3,l}\ \sin(ARG) + c_{y,3,l}\ \cos(ARG)\,] \cdot t^3 \\
&+ \sum_{m}^{1} [\, s_{y,4,m}\ \sin(ARG) + c_{y,4,m}\ \cos(ARG)\,] \cdot t^4
\end{aligned}
\tag{1.10}
$$

$$
\begin{aligned}
s = &+ 94.0t^0 + 3808.65t^1 - 122.68t^2 - 72574.11t^3 + 27.98t^4 + 15.62t^5 \\
&+ \sum_{i}^{33} [\, s_{s,0,i}\, \sin(ARG) + c_{s,0,i}\, \cos(ARG)\,] \cdot t^0 \\
&+ \sum_{j}^{3} [\, s_{s,1,j}\, \sin(ARG) + c_{s,1,j}\, \cos(ARG)\,] \cdot t^1 \\
&+ \sum_{k}^{25} [\, s_{s,2,k}\, \sin(ARG) + c_{s,2,k}\, \cos(ARG)\,] \cdot t^2 \\
&+ \sum_{l}^{4} [\, s_{s,3,l}\, \sin(ARG) + c_{s,3,l}\, \cos(ARG)\,] \cdot t^3 \\
&+ \sum_{m}^{1} [\, s_{s,4,m}\, \sin(ARG) + c_{s,4,m}\, \cos(ARG)\,] \cdot t^4 \\
&- \frac{XY}{2}
\end{aligned}
\tag{1.11}
$$

the parameter t being Julian centuries since J2000 referenced to Terrestrial Time (TT) given by expression $\langle\ ^1\ \rangle$ and ARG (a function of t) constructed from the fundamental arguments of the nutation theory, whose expressions are given by Tab. 1.

The full IAU 2000/2006 series for X, Y, and s are available electronically on the IERS Conventions Center website [3] . An extract for the largest non-polynomial terms in X and Y is given in tables 1 and 1.

The fundamental arguments of nutation theory

In order to compute $ARG(t)$ of equations 1.9, 1.10, and 1.11 the fundamental arguments of nutation theory must be evaluated. Each of the 14 fundamental arguments of nutation theory (Delaunay variables) can be posed as a 4^{th} order polynomial in t, the coefficients of which are provided in Table 1. Terms $F_1 \ldots F_5$ are called lunisolar and terms $F_6 \ldots F_14$ are planetary. Each set of Fourier and Poisson coefficients, $\{s_i,c_i\}_{X,Y,s}$, are associated with a set of 14 multipliers (or expressions), $a_i = \{1, 0, 0, -2, \ldots\}$, one for each fundamental argument. And so, the $ARG(t)$ term associated with $\{s_i,c_i\}_{X,Y,s}$ is computed as

$$
ARG_i(t) = \sum_{k}^{14} a_{i,k} \cdot F_k(t) \tag{1.12}
$$

[3] ftp://tai.bipm.org/iers/convupdt/chapter5, tab5.2a.txt, tab5.2b.txt, and tab5.2d.txt

X

	$s_{x,0,i}$	$c_{x,0,i}$														
1	-6844318.44	1328.67	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	-523908.04	-544.75	0	0	2	-2	2	0	0	0	0	0	0	0	0	0
3	-90552.22	111.23	0	0	2	0	2	0	0	0	0	0	0	0	0	0
...																
	$s_{x,1,j}$	$c_{x,1,j}$														
1307	-3309.73	205833.11	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1308	198.97	12814.01	0	0	2	-2	2	0	0	0	0	0	0	0	0	0
...																
	$s_{x,2,k}$	$c_{x,2,k}$														
1560	2037.98	81.46	0	0	0	0	1	0	0	0	0	0	0	0	0	0
...																

Table 1.1: Series coefficients of X with associated fundamental argument multipliers.

Y

	$s_{y,0,i}$	$c_{y,0,i}$														
1	1538.18	9205236.26	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	-458.66	573033.42	0	0	2	-2	2	0	0	0	0	0	0	0	0	0
3	137.41	97846.69	0	0	2	0	2	0	0	0	0	0	0	0	0	0
...																
	$s_{y,1,j}$	$c_{y,1,j}$														
963	153041.79	853.32	0	0	0	0	1	0	0	0	0	0	0	0	0	0
964	11714.49	-290.91	0	0	2	-2	2	0	0	0	0	0	0	0	0	0
...																
	$s_{y,2,k}$	$c_{y,2,k}$														
1240	120.56	-2301.27	0	0	0	0	1	0	0	0	0	0	0	0	0	0
...																

Table 1.2: Series coefficients of Y with associated fundamental argument multipliers

s

	$s_{s,0,i}$	$c_{s,0,i}$														
1	-2640.73	0.39	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	-63.53	0.02	0	0	0	0	2	0	0	0	0	0	0	0	0	0
3	-11.75	-0.01	0	0	2	-2	3	0	0	0	0	0	0	0	0	0
...																
	$s_{s,1,j}$	$c_{s,1,j}$														
34	-0.07	3.57	0	0	0	0	2	0	0	0	0	0	0	0	0	0
35	1.73	-0.03	0	0	0	0	1	0	0	0	0	0	0	0	0	0
...																
	$s_{s,2,k}$	$c_{s,2,k}$														
37	743.52	-0.17	0	0	0	0	1	0	0	0	0	0	0	0	0	0
...																

Table 1.3: Series coefficients of S with associated fundamental argument multipliers

	t^0	t^1	t^2	t^3	t^4
F_1	485868.249036	1717915923.2178	31.8792	0.051635	-0.00024470
F_2	1287104.793048	129596581.0481	-0.5532	0.000136	0.00001149
F_3	335779.526232	1739527262.8478	-12.7512	-0.001037	0.00000417
F_4	1072260.703692	1602961601.2090	-6.3706	0.006593	-0.00003169
F_5	450160.398036	-6962890.5431	7.4722	0.007702	-0.00005939
F_6	4.402608842	2608.7903141574	0.0	0.0	0.0
F_7	3.176146697	1021.3285546211	0.0	0.0	0.0
F_8	1.753470314	628.3075849991	0.0	0.0	0.0
F_9	6.203480913	334.0612426700	0.0	0.0	0.0
F_{10}	0.599546497	52.9690962641	0.0	0.0	0.0
F_{11}	0.874016757	21.3299104960	0.0	0.0	0.0
F_{12}	5.481293872	7.4781598567	0.0	0.0	0.0
F_{13}	5.311886287	3.8133035638	0.0	0.0	0.0
F_{14}	0.0	0.024381750	0.00000538691	0.0	0.0

Table 1.4: Polynomial coefficients for each of the lunisolar and planetary fundamental arguments.

Chapter 2

Implementation

Clearly much of the work involved in this transformation lyes in the series computation of X, Y, and s. While large sprawling summation do not pose an issue in FORTRAN or C languages, such traditional formulations are not well suited for MATLAB. Thus the main focus of this chapter is to reformulate the calculations of the CIP coordinates using matrix algebra. Such a formulation reduces the notational complexities and provides large scalability improvements.

2.1 Matrix Algebra Formulation

First lets define the number of terms in each of the X,Y, and s series as:

$$\begin{aligned} N_X =& [1306, 253, 36, 4, 1] \\ N_Y =& [962, 277, 30, 5, 1] \\ N_s =& [33, 3, 25, 4, 1] \end{aligned} \tag{2.1}$$

where $sum(N_X) = 1600$, $sum(N_Y) = 1275$, and $sum(N_s) = 66$. Here for example $N_X(1)$ defines the number of t^0 X series terms (non polynomial terms), $N_X(2)$ the number of t^1 X series terms, and so on (see Eq. 1.9, 1.10, and 1.11).

Next lets define the series coefficient vectors $\vec{C}_{X,Y,s}$ and $\vec{S}_{X,Y,s}$ as:

$$
\begin{aligned}
\vec{C_X} &= \begin{bmatrix} c_{x,1} \\ c_{x,2} \\ \vdots \\ c_{x,1600} \end{bmatrix} \quad \vec{S_X} = \begin{bmatrix} s_{x,1} \\ s_{x,2} \\ \vdots \\ s_{x,1600} \end{bmatrix} \\
\vec{C_Y} &= \begin{bmatrix} c_{y,1} \\ c_{y,2} \\ \vdots \\ c_{y,1275} \end{bmatrix} \quad \vec{S_Y} = \begin{bmatrix} s_{y,1} \\ s_{y,2} \\ \vdots \\ s_{y,1275} \end{bmatrix} \\
\vec{C_s} &= \begin{bmatrix} c_{s,1} \\ c_{s,2} \\ \vdots \\ c_{s,66} \end{bmatrix} \quad \vec{S_s} = \begin{bmatrix} s_{s,1} \\ s_{s,2} \\ \vdots \\ s_{s,66} \end{bmatrix}
\end{aligned} \tag{2.2}
$$

and notice no distinction is made between coefficients of varying powers t as such information is maintained by $N_{X,Y,s}$.

Now suppose we wish to evaluate the X,Y, and s series at N times

$$
\vec{t} = \left(t_1, t_2, \ldots, t_N\right) \tag{2.3}
$$

we must first compute each of the 14 fundamental arguments at each time t_i. Let $F\left(\vec{t}\right)$ be the matrix of having evaluated $F_k(t_i)$ for $k = \{1 \ldots 14\}$ and $i = \{1 \ldots N\}$. Thus we have

$$
F\left(\vec{t}\right) = \begin{pmatrix} F_{1,1} & F_{1,2} & \cdots & F_{1,14} \\ F_{2,1} & F_{2,2} & \cdots & F_{2,14} \\ \vdots & \vdots & \ddots & \vdots \\ F_{N,1} & F_{N,2} & \cdots & F_{N,14} \end{pmatrix}_{N \times 14} \tag{2.4}
$$

where the i^{th} row of F contains the 14 fundamental arguments evaluated at time t_i.

Upon inspection, the definition of $ARG(t)$ (Eq. 1.12) is simply a dot product of the expression vector and the fundamental arguments at some time t. Consider momentarily the X series and note that for each time t_i there are 1600 ARG terms. That is, at each time t_i, each series coefficient pair $\{c_j, s_j\}_{X,Y,s}$ is associated with it's own ARG_j term. Therefore

expression matrices $A_{X,Y,s}$ are defined containing the multipliers for each coefficient pair:

$$A_X = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & \cdots & & & & & & & & & & & & \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & \cdots & & & & & & & & & & & & \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & \cdots & & & & & & & & & & & & \end{pmatrix}_{1600\times 14} \tag{2.5}$$

where the jth row corresponds to coefficients $\{c_j, s_j\}_{X,Y,s}$. Likewise, the expression matrices A_Y, and A_s are defined accordingly. And, thus the ARG dot products at each time t_i for X, Y, and s are formed via matrix-matrix multiplication as:

$$ARG = (A\, F^T)^T \tag{2.6}$$

So for example, ARG_X would be

$$(A_X F^T)^T = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,1600} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,1600} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N,1} & \phi_{N,2} & \cdots & \phi_{N,1600} \end{pmatrix}_{N\times 1600} \tag{2.7}$$

where the ith row corresponds to time t_i. Similarly ARG_Y has dimensions $N \times 1275$ and ARG_s has dimensions $N \times 66$. It's worth computing a few $\phi_{i,j}$ by hand to ensure that the ARG matrices are fully understood.

There is one more matrix required to compute the series summations. This matrix makes sure that each sin and cos term is multiplied by the correct power of t. Therefore, we construct a matrix T having the same dimensions as ARG where the ith row contains the appropriate powers of t_i such that

$$T_i = \begin{pmatrix} t_i^0 & \ldots & t_i^0 & t_i^1 & \ldots & t_i^1 & t_i^2 & \ldots & t_i^2 & t_i^3 & \ldots & t_i^3 & t_i^4 & \ldots & t_i^4 \end{pmatrix} \tag{2.8}$$

Clearly, according to the definitions of $N_{X,Y,s}$ (Eq. 2.1), T_i corresponding with the X series would contain 1306 t_i^0 terms, 253 t_i^1 terms, and so on.

We are now ready to compute the X, Y, and s series. In general, each series is simply computed as

$$P + [\cos(ARG) \circ T]\, \vec{C} + [\sin(ARG) \circ T]\, \vec{S} \tag{2.9}$$

where the $\circ$ indicates element-wise matrix multiplication (*not* matrix-matrix multiply) where $(A \circ B)_{ij} = (A)_{ij}(B)_{ij}$ [1] [2] . So the X series for $\vec{t}$ would be computed as

$$X(\vec{t}) = P_X(\vec{t}) + [\cos(ARG_X) \circ T_X]\, \vec{C}_X + [\sin(ARG_X) \circ T_X]\, \vec{S}_X \tag{2.10}$$

having dimensions $N \times 1$ where $P_{X,Y,s}(\vec{t})$ are the polynomial expressions defined by equations 1.9, 1.10, and 1.11 evaluated at each time $t_i \in t$.

2.2 MATLAB Implementation Details

Now that a matrix formulation is in place it is rather trivial to implement in MATLAB. At this point, making sure each function uses the correct time reference system and outputs the correct units will be the most challenging aspect.

As demonstrated by the generalized series summation (Eq. 2.9) a single generalized algorithm can be constructed that agnostically computes the X,Y,s series given the appropriate matrices as discussed in the previous section. The following code listing does just that:

```
1         function seriesSum = seriesImpl(t,F,A,N,c,s)
2
3             % compute powers of t for Poisson terms [Nx{1600,1275,66}]
4             T = [                                   ...
5                     t(:,ones(1,N(1))).^0 ...
6                     t(:,ones(1,N(2))).^1 ...
7                     t(:,ones(1,N(3))).^2 ...
8                     t(:,ones(1,N(4))).^3 ...
9                     t(:,ones(1,N(5))).^4 ...
10            ];
11
12            % compute the argument [Nx{1600,1275,66}]
13            ARG = (A*F')';
14
15            % the sum of the series at each time [Nx1]
16            seriesSum = (sin(ARG).*T)*s + (cos(ARG).*T)*c;
17
18         end
```

Equipped with engine for computing each of the series terms, X,Y, and s can be easily computed. The following listing is an object method belonging to the IERSLib.m class. The IERSLib.m has internally defined the relevant static quantities (mostly matrices and a few constants) necessary for the computation of the transformation

[1] This is equivalent in matlab to $A .* B$

[2] $A \circ B$ is also known as a *Hadamard* product

```
function [X,Y,s] = XYs(this,t)

    % input:
    %
    %     t  [Nx1]  - fMJD referenced to TT
    %

    % make sure t is a column vector
    t = t(:);

    % adjust time
    dt = IERSLib.MJD_REF_EPOCH - IERSLib.DJ00;

    % compute the time in julian centeries
    t = ( dt + t ) ./ IERSLib.DJC;

    % compute basis vectors in t
    tt = [ t.^0   t.^1   t.^2   t.^3   t.^4   t.^5 ];

    % compute the fundamental args of nutation at each time t
    F = IERSLib.F(t);

    % compute X  [microarcseconds]
    X = tt * this.Px + ...
        IERSLib.seriesImpl(t,F,this.Ax,this.Nx,this.Cx,this.Sx);

    % compute Y  [microarcseconds]
    Y = tt * this.Py + ...
        IERSLib.seriesImpl(t,F,this.Ay,this.Ny,this.Cy,this.Sy);

    % compute s  [microarcseconds]
    s = tt * this.Ps + ...
        IERSLib.seriesImpl(t,F,this.As,this.Ns,this.Cs,this.Ss);

    % convert to arcseconds
    X = X./1e6;  Y = Y./1e6;  s = s./1e6;

    % convert to radians
    X = X.*IERSLib.AS2R;  Y = Y.*IERSLib.AS2R;  s = s.*IERSLib.AS2R;

    % tidy up s
    s = s - (X.*Y)./2;
end
```

It is certainly worth including the implementation of ERA. Typically, the Celestial to Terrestrial transformation is very sensitive to errors in $\Delta UT1$ via the Earth Rotation Angle. Even with very accurate values of $\Delta UT1$ the implementation is subject to roundoff errors associated with how the time computations are handled. The following code listing computes ERA consistent with equivalent SOFA implementations at the 10^{-15} level

```
function theta    = ERA(fMJD_UTC,du)
    %
    % [E]arth [R]otation [A]ngle
    %
    %  input:
    %
    %    t      [Nx1] — fMJD times (referenced to UTC)
    %
    %    du     [Nx1] — UTC—UT1 offset
    %
    %  output:
    %
    %    theta [Nx1] — era at time specified by t [radians]
    %
    %  NOTE:
    %
    %    MJD reference epoch: 2400000.5
    %
    %    The direction is CIRS -> TIRS
    %

    % compute whole julian date reference to utc
    JD_UTC = floor(fMJD_UTC) + IERSLib.MJD_REF_EPOCH;

    % compute fractional part of day referenced to ut1
    tut    = (fMJD_UTC — floor(fMJD_UTC)) + du./86400;

    % compute time in julian centeries
    t = tut + (JD_UTC — IERSLib.DJ00);

    % Fractional part of T (days).
    f = rem(tut, 1.0) + rem(JD_UTC,1.0);

    % Earth rotation angle at this UT1
    theta = 2 * pi *(f + 0.7790572732640 + 0.00273781191135448 * t);

    % normalize between zero and 2 * pi
    theta = rem(theta,2*pi);  theta(theta<0) = theta(theta<0)+2*pi;

end
```

Finally, for the sake of completeness the code listing for computing the fundamental arguments of nutation theory consistent with the matrix algebra formulation is provided. Note that the vectors $F_1 \ldots F_{14}$ are the coefficient vectors defined by Table 1

```
function f = F(t)
    % input:
    %
    %    t [Nx1] - TDB, time in Julian centuries since J2000.0
    %
    % output:
    %
    %    f [Nx14] - lunisolar(5) and planetary(9) args in radians
    %
    %  NOTES:
    %
    %    Ref: IERS Conventions 2010, ch5, pp 67 & 68
    %
    %    Ref: www.iausofa.org/2012_0301_C/FundArgs.html
    %

    % make sure t is a column vector
    t = t(:);

    % form basis vectors
    tt = [ t.^0  t.^1  t.^2  t.^3  t.^4 ];

    % compute the lunisolar terms
    lunisolar = [tt*F1, tt*F2, tt*F3, tt*F4, tt*F5];

    % tidy up (convert to arcseconds then to radians)
    lunisolar = rem(lunisolar,IERSLib.TURNAS).*IERSLib.AS2R;

    % compute the planetary terms
    planetary = [
                 tt*F6 , tt*F7 , tt*F8 , tt*F9 ,  ...
                 tt*F10, tt*F11, tt*F12, tt*F13,  ...
    ];

    % tidy up, normalize between 0 and 2*pi
    planetary = rem(planetary,2*pi);

    % make sure any negative radians get "flipped"
    planetary(planetary<0) = planetary(planetary<0)+2*pi;

    % that's all folks (f in radians)
    f = [lunisolar planetary tt*F14];
end
```

The remaining quantities such as $Q(t)$, W, and so on can be computed directly as

presented. Note that s' need be computed in radians.

2.3 Performance

It is commonly the case that FORTAN and C compiled implementations outperform MATLAB scripts. However in recent years MATLAB has eliminated many bottlenecks and introduced advanced features under the hood such as Just In Time (JIT) compilation of *.m* script files that alleviate many of the poor performance issues of yesteryear. It is also essential to understand how "vectorization" plays a crucial role in MATLAB performance. The MATLAB Interpreter (i.e. the program that goes through your english *.m* file and translate it into something the machine can understand) is quite slow. The idea here is to invoke the interpreter as few times as possible. To do this one must speak to interpreter in it's native tongue by posing problems vector operations rather than scalar manipulation. For example, consider two vectors a and b such that

$$a = b = \begin{pmatrix} 1 & 2 & \dots & 10000 \end{pmatrix} \tag{2.11}$$

the dot product written as a bunch of scalar operations:

```
1 c = 0;
2 for i = 1:numel(a)
3     c = c + a(i)*b(i);
4 end
```

yields poor performance (≈ 0.004s) while the "vectorized" version:

```
1 c = a*b'
```

is roughly 40 times faster (≈ 0.00009s)!

It is easy to extrapolate from this simple example that the sprawling summations in the definitions of X, Y and s (Eq. 1.9, 1.10, and 1.11) are not well suited for the MATLAB scripting environment. Thus it is vital that the computation of the CIP coordinates be reformulated in a matrix algebra fashion in order for a MATLAB implementation to be successful. The additional benefit of this reformulation is *scalability*. It is possible to compute the CIP coordinates at many times t_i in a single operation rather than looping over an array of times one-by-one. That is, it is possible to compute the single quantity $X(\vec{t})$ rather than many independent $X(t_i)$. For large numbers of coordinate transformations this is a huge advantage. Figure 2.1 compares the MATLAB matrix algebra approach to that of the traditional FORTRAN implementation for many points.

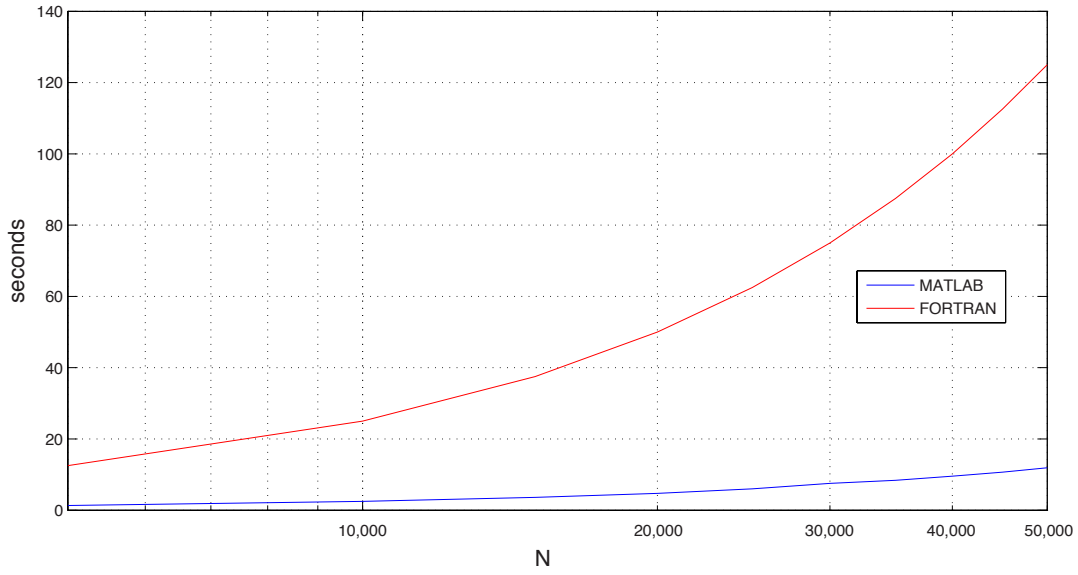

Figure 2.1: *Performance comparison.* Here traditional FORTRAN computation of many independent $X(t_i)$ (XYS2000A.F) is compared to the MATLAB calculation of $X(\vec{t})$. Notice that the independent computation of CIP coordinates at 50,000 times t_i requires more than 120 seconds while the MATLAB implementation clocks in just under 17 seconds

.

2.4 Test Case Tutorial

One of the biggest difficulties in performing the celestial to terrestrial transformation (and visa versa) is making sure each input and output is of the correct unit and/or time reference. Here a "walkthrough" is provided for computing this transformation that is in part to demonstrate the functionality of IERSLib.m, and it's dancing partner EOPLib.m, while also providing valuable test data necessary for debugging custom code.

In order to start this process we must first define the exact time of the coordinates to be transformed (referenced to UTC) and then obtain EOP information consistent with the time. Such information is provided in the following listing

```
% set the date [UTC]
MJD  = 53101;

% set the time [UTC]
hour = 7; min = 51;  sec = 28.386009;

% compute the seconds of day
sod  = hour*3600 + min*60 + sec;

% compute the date+time
fMJD_UTC = MJD +  sod/86400;

% EOP information [arcseconds, sec, sec]
xp = -0.140682;  yp =  0.333309;  du = -0.439962; dt = 32;
```

where x_p and y_p are the polar motion coordinates in arcseconds, du is the $UT1 - UTC$ offset ($\Delta UT1$) in seconds, and dt is the $UTC - TAI$ offset (ΔAT) in seconds[3,4].

Next, the XYZ coordinate, in kilometer, to be transformed is defined precisely in both the ITRS and GCRS frames as

```
% Vallado et al. 2006, AIAA NOTE: using IERS 2003  [km]
X_itrs = [-1033.4793830, 7901.2952754, 6380.3565958]';

% dx,dy = 0                                          [km]
X_gcrs = [5102.5089592, 6123.0114033, 6378.1369247]';
```

At this point it is important to note that proceeding code listings are developments for converting GCRS $\rightarrow$ ITRS. That is we will start with a coordinate in the Geocentric Reference System, X_{gcrs} and transform it into the International Terrestrial Reference System, X_{ITRS}.

[3]USNO provide estimates of x_p,y_p and $\Delta UT1$ at http://maia.usno.navy.mil/ser7/ser7.dat
[4]USNO provide estimates of ΔAT at http://maia.usno.navy.mil/ser7/leapsec.dat

In preparation, using ΔAT we can compute the time referenced to Terrestrial Time (TT), and convert the polar motion coordinates to radians

```
% fractional mjd referenced to terestrial time
t_TT  = t + (dt + IERSLib.TT_OFFSET)/86400;

% convert xp and yp to radians
xp = xp.*IERSLib.AS2R;  yp = yp.*IERSLib.AS2R;
```

where the $TAI - TT$ offset is a constant 32.184 seconds and $AS2R = \pi/648000$. Now from here we can compute the Terrestrial Intermediate Origin (TIO) locator s' in radians as

```
% compute the TIO locator [radians]
sp = IERSLib.SP00(t_TT);
```

using Eq 1.3 where internally Terrestrial Time, t_TT, is converted to Julian centuries

```
% compute the julian centuries since the J2000 reference epoch
t = ( ( IERSLib.MJD_REF_EPOCH - IERSLib.DJ00) + t_TT) / IERSLib.DJC;
```

With the TIO locator in hand the polar motion matrix, $W(t)$ of Eq. 1.1, can be computed as

```
% comput the polar motion matrix
P = IERSLib.POM00(xp,yp,sp);
```

where each of the three inputs are in radians. For debugging purposes the values of P are

$$P = \begin{pmatrix} 0.999999999999767 & -0.000000000009712 & -0.000000682045583 \\ 0.000000000008610 & 0.999999999998694 & -0.000001615927632 \\ 0.000000682045583 & 0.000001615927632 & 0.999999999998462 \end{pmatrix} \tag{2.12}$$

The next Task is to compute the Earth Rotation Angle so as to formulate the rotation matrix, $R(t)$ of Eq. 1.1. To this effect we have

```
% compute the earth rotation angle [radians]
theta_ERA = IERSLib.ERA(t,du);
```

where input du is the ΔUT1 offset in seconds (always within ± 0.9s) and the resulting rotation angle is normalized within $[0, 2\pi]$

$$\theta_{ERA} = 5.458609437785398 \tag{2.13}$$

where the previous code listing provides additional ERA implementation details to ensure proper handling of roundoff errors

The Celestial Intermediate Pole (CIP) coordinates are computed as

```
% compute the unit vectors to CIO
[X,Y,s] = this.XYs(t_TT);
```

where X, Y, and s are output in radians as unit vectors to the Celestial Intermediate Origin (CIO) having values

$$\begin{aligned} X &= 3.904295831038141 \times 10^{-4} \\ Y &= 3.526485630691542 \times 10^{-5} \\ s &= -1.461649816564349 \times 10^{-8} \end{aligned} \tag{2.14}$$

The Celestial to Terrestrial Intermediate transformation matrix, $Q(t)$ of Eq. 1.1, is computed as

```
% compute the CIO xformation matrix (i.e. Q(t))
C2T = IERSLib.C2I(X,Y,s);
```

can be constructed directly from Eq. 1.6 and has resulting values

$$C2T = \begin{pmatrix} 0.999999923782367 & 0.000000007732276 & -0.000390429583619 \\ -0.000000021500719 & 0.999999999378195 & -0.000035264850600 \\ 0.000390429583104 & 0.000035264856307 & 0.999999923160562 \end{pmatrix} \tag{2.15}$$

Finally, having each of the required quantities ready, the GCRS $\rightarrow$ ITRS transformation matrix can be calculated. The following code listing provides the final auxiliary calculation

```
% put it all together ...
GC2IT = P * IERSLib.R3(theta_ERA) * C2T;
```

where the X_{itrs} state is calculated as

```
% transform the coordinate from GCRS to ITRS
X = GC2IT*X_gcrs;
```

having a values

$$X = \begin{pmatrix} -1033.479392368547 & 7901.295274652139 & 6380.356595221698 \end{pmatrix} \tag{2.16}$$

with RMS difference of 5.45 [mm] from the expected result.

Appendix A

Time Reference Systems

Timekeeping means following an agreed recipe for measuring time, using some natural clock as a basis. The most practical phenomena for this purpose are rotations and oscillations, and for most of history the Earths rotation was the best available timekeeper. Because time was synonymous with the cycle of day and night, the units used to express time intervals reflect that choice: days, divided into hours, minutes and seconds. For larger intervals, other astronomical phenomena play a role, in particular the orbital periods of the Earth and Moon.

Repeated attempts in the 19th century to model the motion of solar system objects, especially the fast-moving Moon, were only partially successful. Each new theory, while reproducing existing data accurately, would soon start to diverge from observation. The pragmatic solution was to create a tautology by regarding the theories as clocks in their own right; in effect the clocks hands were the bodies of the solar system, reading out ephemeris time. However, once man-made clocks became sufficiently accurate, suspicions that irregularities in Earth rotation had all along been the limiting factor were confirmed, and laboratory time scales created by such clocks - nowadays atomic, before that quartz took over as the primary timekeeping standard.

The familiar *Gregorian calendar date*, consisting of year, month and day, is designed to keep more or less in step with the tropical year, the year of the seasons, which has a length of about 365.2422 days. Its predecessor, the Julian calendar, approximated the tropical year by using a basic 365-day year and introducing an extra day (February 29) in every fourth year, giving an average year length of 365.25 days. Astronomers still use the latter, the terms Julian year and Julian century meaning exactly 365.25 and 36525 days respectively. The Gregorian calendar provided a better approximation to the tropical year by dropping three such leap years in each 400 years, to give an average of 365.2425 days. The rule is that century years must be divisible by 400 to be a leap year; other years are leap years if they are divisible by 4. So the year 2000 was a leap year, but 2100 will not be a leap year.

A.1 Julian Date

For many purposes, calendar date is inconvenient: what is needed is a continuous count of days. For this purpose the system of Julian day number can be used. JD zero is located about 7000 years ago, well before the historical era, and is formally defined in terms of Greenwich noon;1 for example Julian day number 2449444 began at noon on 1994 April 1. Julian date (JD) is the same system but with a fractional part appended; JD 2449443.5 was the midnight on which 1994 April 1 commenced. Because of the unwieldy size of Julian Dates and the awkwardness of the half-day offset, it is accepted practice to remove the leading 24 and the trailing .5, producing what is called the Modified Julian Date:

$$MJD = JD - 2400000.5. \tag{A.1}$$

Thus 1994 April 1 commenced at MJD 49443.0. MJD is often used in computer applications, rather than JD itself. The use of MJD also reduces exposure to rounding errors.

A.2 Time Scales

Calculations in any scientific discipline may involve precise time, but what sets astronomy apart is the number and variety of time scales that have to be used. There are several reasons for this: astronomers must continue to deal with the very phenomena that lie behind obsolete time scales, in particular the rotation of the Earth and the motions of the planets; as new time scales have been introduced, continuity with the past has been preserved, leaving in the various astronomical time scales a fossil record of former offsets and rates; and in astronomical applications the physical context of the clock matters, whether it is on Earth, moving or stationary, or on a spacecraft.

There are five essential time reference systems required for the Celestial to Terrestrial transformation:

- TAI (International Atomic Time): the official timekeeping standard.
- UTC (Coordinated Universal Time): the basis of civil time.
- UT1 (Universal Time): based on Earth rotation.
- TT (Terrestrial Time): used for solar system ephemeris look-up.
- TDB (Barycentric Dynamical Time): that keeps in step with TT on average

The transformation from one time scale to the next can take a number of forms. In some cases, for example TAI to TT, it is simply a fixed offset. In others, for example TAI to UT1, it is an offset that depends on observations and cannot be predicted in advance (or only partially). Some time scales, for example TT and Geocentric Coordinate Time (TCG), are linearly related, with a rate change as well as an offset. Others require a 4-dimensional space-time transformation.

A.2.1 Atomic Time: TAI

The unit of proper time is the SI second, defined as the duration of 9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the caesium 133 atom, the latter being at 0 K and in zero magnetic field. The duration was chosen to match the existing astronomical time scale, and is consequently a fossil of the solar day length during the 19th century. The SI second is the unit of TAI and UTC.

TAI is a laboratory time scale, independent of astronomical phenomena apart from having been synchronized to solar time when first introduced (at the start of 1958). It is realized through a changing population of about 200 high-precision atomic clocks held at standards institutes in various countries. There is an elaborate process of continuous intercomparison, leading to a weighted average of all the clocks involved. TAI is close to proper time for an observer on the geoid, and is an appropriate choice for terrestrial applications where continuity through UTC leap seconds is a requirement. It is not disseminated directly as a time service, but can easily be realized from GPS or UTC.

A.2.2 Solar Time: UT1 and UTC

UT1 (or plain UT) is the modern equivalent of mean solar time, and is really an angle rather than time in the physics sense. Originally defined in terms of a point in the sky called the fictitious mean Sun, UT1 is now defined through its relationship with Earth rotation angle (formerly through sidereal time). Because the Earths rotation rate is slightly irregular - for geophysical reasons - and is gradually decreasing, the UT1 second is not precisely matched to the SI second. This makes UT1 itself unsuitable for use as a time scale in physics applications. However, some applications do require UT1, such as pointing a telescope or antenna at a celestial target, delay calculations in interferometers, and diurnal aberration, parallax and Doppler corrections.

UTC is a compromise between the demands of precise timekeeping and the desire to maintain the current relationship between civil time and daylight. Since its introduction in 1960, UTC has been kept roughly in step with UT1 by a variety of adjustments agreed in advance and then carried out in a coordinated manner by the providers of time services - hence the name. Though rate changes were used until 1972, since then all such adjustments have been made by occasionally inserting a whole second, called a leap second, a procedure that can be thought of as stopping the UTC clock for a second to let the Earth catch up. Leap seconds are discussed below.

To obtain UT1 starting from UTC, it is necessary to look up the value of

$$\Delta UT1 = UT1 - UTC \tag{A.2}$$

for the date concerned in tables published by the International Earth Rotation and Reference Systems Service (IERS); this is then added to the UTC. The quantity $\Delta UT1$, which

typically changes by 1-2 ms per day, can be obtained only by observation, principally very long baseline interferometry (VLBI) using extragalactic radio sources, though seasonal effects are present and the IERS listings are able to predict some way into the future with adequate accuracy for most applications[1].

A.2.3 Leap Seconds

Leap seconds are introduced as necessary to keep $UT1 - UTC$ in the range ± 0.9 seconds. The decision is made by international agreement, and announced several months in advance by the IERS. Leap seconds occur usually at the end of December or June. Because on the average the solar day is now 1-2 ms longer than the nominal 86,400 SI seconds, accumulating to 1s over a period of 18 months to a few years, leap seconds are in practice always positive; however, provision exists for negative leap seconds if needed. Each time a leap second is introduced, the offset $\Delta AT = TAI - UTC$ changes by exactly 1s. Up-to-date information on TAI - UTC is available from the IERS[2].

Note that UTC has to be expressed as hours, minutes and seconds (or at least in seconds in a given day) if leap seconds are to be taken into account in the correct manner. In particular, it is inappropriate to express UTC as a Julian Date, because there will be an ambiguity during a leap second - so that for example 1994 June 30 23h 59m 60s.0 and 1994 July 1 00h 00m 00s.0 would both come out as MJD 49534.00000 - and because subtracting two such JDs would not yield the correct interval in cases that contain leap seconds.

A.2.4 TT

Terrestrial time, TT (called TDT between 1984 and 2000), is the theoretical time scale for clocks at sea-level: for practical purposes it is tied to TAI through the fixed formula:

$$TT = TAI + 32.184s \tag{A.3}$$

Calculating TT from UTC, rather than TAI, requires leap seconds to be taken into account (i.e. ΔAT).

[1] USNO provide estimates of $\Delta UT1$ at http://maia.usno.navy.mil/ser7/ser7.dat
[2] USNO provide estimates of ΔAT at http://maia.usno.navy.mil/ser7/leapsec.dat