Relativistic time scales and time transformations for *Gaia*

Sergei A. Klioner

Lohrmann Observatory, Technische Universität Dresden, 01062 Dresden, Germany

GAIA-CA-TN-LO-SK-012-5
Issue 5, February 7, 2025

The goal of this note is to summarize the relativistic time scales and the transformations between them relevant for *Gaia* and to give efficient algorithms to calculate them. The note describes the theoretical framework used in the Java implementation for *Gaia*.

Issue 3 corrects several misprints and adds a discussion of the additional features of the Java code appeared after the integration into the GaiaTools.

Issue 4 corrects a few misprints in the text.

Issue 5 includes the values of the relevant constants used in *Gaia* as well as updates the text to the actual development state of the corresponding software.

I. TIME SCALES RELEVANT FOR *GAIA*

Let us first list the time scales that are relevant for *Gaia* and summarize their properties and the intended use in *Gaia*:

– On-board Time (`OBT`) is the reading of the physical clock on board of *Gaia*. The physical clock of *Gaia* is a free running oscillator with no apriori relation to any times scales like `TCB` or others.

  Use in Gaia: `OBT` is used in the time tags (labels) of raw data.

– On-board Modified Time (`OBMT`) is `OBT` corrected for possible clock resets (switching the *Gaia* clock off and then on, as it can happen during safe modes of the satellite, resets the clock counter). The details of the definition of `OBMT` and the relation between `OBT` and `OBMT` is described in [2]. The relation between `OBMT` and other time scales is the subject of the *Gaia* clock calibration: Low-Accuracy Time Transformation (LATT) and High-Accuracy Time Transformation (HATT). These calibrations are computed using special clock synchronization observations (the so-called time packets) and are not part of this note. Some details of the *Gaia* clock calibrations can be found in [9, 10]. An exhaustive description will be published elsewhere.

  Use in Gaia: `OBMT` is used as an intermediate step in the process of reparametrizing of the *Gaia* data from `OBT` to `TCB` (see, [2] for further discussion).

– *Gaia* Time (`TG`) is the ideal proper time of *Gaia*: `TG` is the reading of a hypothetical ideal clock moving together with the center of mass of *Gaia*. The relation of `TG` and `TCB` is discussed in Section III below.

  Use in Gaia: `TG` is an intermediate step in the process of reparametrizing of the *Gaia* data from `OBT` to `TCB`. It can be used also to improve the dynamical model of rotational motion of the *Gaia* satellite.

– Barycentric Coordinate Time (TCB) is the coordinate time of Barycentric Celestial Reference System (BCRS) of the International Astronomical Union (IAU). A detailed discussion of the BCRS can be found in [16].

  Use in Gaia: TCB is the main time scale in the *Gaia* data processing. The final *Gaia* astrometric catalog represents a materialization of the BCRS in the sense that all coordinates, velocities and distances in the *Gaia* catalog are defined in the BCRS, and all moments of time are given in TCB.

– Barycentric Dynamical Time (TDB) is a conventional linear function of TCB:

$$\mathtt{TDB} = \mathtt{TCB} - L_B \times (JD_{\mathtt{TCB}} - T_0) \times 86400 + \mathtt{TDB}_0 \tag{1}$$

  with $T_0 = 2443144.5003725$, $L_B = 1.550519768 \times 10^{-8}$, $\mathtt{TDB}_0 = -6.55 \times 10^{-5}$ s. This transformation is given by a Resolution of the IAU 2006. TDB is the time scale used by e.g., the JPL ephemerides and the INPOP version used by the flight dynamics group at ESOC (an export version of the INPOP is used by the ESOC). The values of T_0, L_B are chosen in order to effectively eliminate the linear drift between TDB and TT at the geocenter when the transformation is computed using the JPL ephemeris DE405. The constant term $\mathtt{TDB}_0$ is chosen to provide reasonable consistency with the widely used TDB − TT formula of Fairhead and Bretagnon [6].

  Use in Gaia: TDB is used as a conventional substitute of TCB in some DPAC-external data sets. Currently, examples of the use of TDB are: (1) the ESOC orbit of *Gaia* will be initially parametrized by TDB, (2) the transformation between TCB and TCG at the geocenter is computed via the transformation between TDB and TT at the geocenter delivered by the INPOP team. The data sets parametrized by TDB will be converted to TCB for further use in the *Gaia* data processing chain.

– Geocentric Coordinate Time (TCG) is the coordinate time of Geocentric Celestial Reference System (GCRS) of the International Astronomical Union (IAU). The relation between TCG and TCB will be discussed in Section II below.

  Use in Gaia: TCG is used as an intermediate step to compute TCB moments of reception of a time packet originally labelled by UTC (or GPS time). It is only used as intermediate step for the OBMT calibration.

– TT is a conventional linear function of TCG:

$$\mathtt{TT} = \mathtt{TCG} - L_G \times (JD_{\mathtt{TCG}} - T_0) \times 86400 \tag{2}$$

  with $T_0 = 2443144.5003725$, $L_G = 6.969290134 \times 10^{-10}$. The coefficients of the linear function are chosen is such a way that TT is as close as possible to proper time of an observer situated on the rotating geoid. This definition is given by the IAU 2000 [16].

  Use in Gaia: TT is used as an intermediate step to compute TCB moments of reception of a time packet originally labelled by UTC (or GPS time). It is only used as intermediate step for the OBMTcalibration.

– International atomic time (TAI) is a high-precision atomic coordinate time standard representing a practical realization of TT. TAI is related to TT as

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

Use in Gaia: TAI is used as an intermediate step to compute TCB moments of reception of time packets originally labelled by UTC (or GPS time). It is only used as intermediate step for the OBMTcalibration.

– Coordinated Universal Time (UTC) is a time standard based on International Atomic Time (TAI). UTC differs from TAI by an integer number of leap seconds, so that UT1 − UTC stays smaller than 0.9 s in absolute value, UT1 being the rotation angle about the Earth pole (UT1 can be regarded as a time determined by the rotation of the Earth). UTC and TAI are related as

$$\texttt{TAI} = \texttt{UTC} + dAT(\texttt{UTC}), \tag{4}$$

where dAT is the number of leap seconds up to given moment of UTC. The leap seconds are announced in the IERS Bulletins C (http://www.iers.org). The whole history of function dAT(UTC) can be found, e.g., in http://hpiers.obspm.fr/eoppc/bul/bulc/UTC-TAI.history. These values are also included in *Gaia* Parameter Database as GaiaParam.Nature.TAIMINUTC_CONSTANT.

Use in Gaia: UTC is used to label (time stamp) time packets. It is only used in the raw time packets and as intermediate step for the OBT calibration.

– GPS time is the atomic time scale implemented by the atomic clocks in the GPS ground control stations and the GPS satellites themselves. GPS time was zero at 0h 6-Jan-1980 and since it is not perturbed by leap seconds. GPS time is now ahead of UTC by the corresponding number of leap seconds. The relation to TAI is fixed:

$$\texttt{TAI} = \texttt{GPS} + 19 \text{ s}. \tag{5}$$

Use in Gaia: GPS time is not used in the *Gaia* data processing and described here for completeness. It could appear in the *Gaia* data processing chain, e.g., in the time labels of the time packets (instead of UTC), but the ESA ground stations use UTC. One more possible application of GPS time was the data processing for Nano-JASMINE [11] (but this project was eventually abandoned).

II. TRANSFORMATION BETWEEN TCB AND TCG

The transformation between t = TCB and T = TCG is a part of the standard transformation between BCRS and GCRS coordinates as defined by the IAU (see IAU 2000 Resolutions at http://www.iau.org/administration/resolutions/general_assemblies/ and [16]):

$$T = t - \frac{1}{c^2}\left[A(t) + v^i_E r^i_E\right] + \frac{1}{c^4}\left[B(t) + B^i(t) r^i_E + B^{ij}(t) r^i_E r^j_E + C(t,\mathbf{x})\right] + O(c^{-5}), \tag{6}$$

$$X^a = \delta_{ai}\left[r^i_E + \frac{1}{c^2}\left(\frac{1}{2} v^i_E v^j_E r^j_E + w_{\text{ext}}(\mathbf{x}_E) r^i_E + r^i_E a^j_E r^j_E - \frac{1}{2} a^i_E r^2_E\right)\right] + O(c^{-4}), \tag{7}$$

where

$$\frac{d}{dt}A(t) = \frac{1}{2}v_E^2 + w_{\rm ext}(\mathbf{x}_E), \tag{8}$$

$$\frac{d}{dt}B(t) = -\frac{1}{8}v_E^4 - \frac{3}{2}v_E^2 w_{\rm ext}(\mathbf{x}_E) + 4v_E^i w_{\rm ext}^i(\mathbf{x_E}) + \frac{1}{2}w_{\rm ext}^2(\mathbf{x}_E), \tag{9}$$

$$B^i(t) = -\frac{1}{2}v_E^2 v_E^i + 4w_{\rm ext}^i(\mathbf{x}_E) - 3v_E^i w_{\rm ext}(\mathbf{x}_E), \tag{10}$$

$$B^{ij}(t) = -v_E^i \delta_{aj} Q^a + 2\frac{\partial}{\partial x^j}w_{\rm ext}^i(\mathbf{x}_E) - v_E^i \frac{\partial}{\partial x^j}w_{\rm ext}(\mathbf{x}_E) + \frac{1}{2}\delta^{ij}\dot{w}_{\rm ext}(\mathbf{x}_E), \tag{11}$$

$$C(t,\mathbf{x}) = -\frac{1}{10}r_E^2(\dot{a}_E^i r_E^i). \tag{12}$$

Here x_E^i, v_E^i, and a_E^i are the barycentric position, velocity and acceleration vectors of the Earth, the dot stands for the total derivative with respect to t, and

$$Q^a = \delta_{ai}\left[\frac{\partial}{\partial x_i}w_{\rm ext}(\mathbf{x}_E) - a_E^i\right]. \tag{13}$$

The external potentials, $w_{\rm ext}$ and $w_{\rm ext}^i$, are given by

$$w_{\rm ext} = \sum_{A \neq E} w_A,$$
$$w_{\rm ext}^i = \sum_{A \neq E} w_A^i\,, \tag{14}$$

where A enumerates solar system bodies and E stands for the Earth. For the purposes of time transformations for *Gaia* it is sufficient to consider solar system as a system of N mass monopoles (bodies characterized by their masses only; no further structure of gravitational field, e.g. quadrupole, is taken into account). In this case one has

$$w_A(\mathbf{x}) = \frac{GM_A}{|\mathbf{x} - \mathbf{x}_A|} + \mathcal{O}(c^{-2}),$$
$$w_A^i(\mathbf{x}) = w_A(\mathbf{x})\, v_A^i\,, \tag{15}$$

where M_A is the mass of body A, $\mathbf{x}_A$ and $\mathbf{v}_A$ are the position and velocity of body A.

A. Transformation between TCB and TCG at a given spatial point

For *Gaia* it is sufficient to consider the transformation between TCB and TCG only within a geocentric sphere with radius $|\mathbf{r}_E| < 2 \times 10^9$ m (this sphere contains the *Gaia* satellite at any moment of time). On the other hand, one can expect that typically the transformations will be applied at the locations of ground stations so that $|\mathbf{r}_E| < 6.4 \times 10^6$ m. Let us give the estimates of the maximal values of the position-dependent terms in (6) for these two cases. Since the time resolution of *Gaia* is worse than 1 ns, it is clear from the values in Table I that it is sufficient to take into account only position-dependent terms of order c^{-2} and neglect terms proportional to B^i, B^{ij} and C.

term\region	$\lvert\mathbf{r}_E\rvert < 6.4\times10^6$ m	$\lvert\mathbf{r}_E\rvert < 2\times10^9$ m
$-\frac{1}{c^2}\,v^i_E r^i_E$	2.2×10^{-6} s	6.8×10^{-4} s
$+\frac{1}{c^4}\,B^i(t)r^i_E$	5.4×10^{-14} s	1.7×10^{-11} s
$+\frac{1}{c^4}\,B^{ij}(t)r^i_E r^j_E$	1.5×10^{-18} s	1.4×10^{-13} s
$+\frac{1}{c^4}\,C(t,\mathbf{x})$	4.6×10^{-24} s	1.4×10^{-16} s

TABLE I: Estimations of location-dependent terms in the TCB-TCG transformation.

On the other hand, transformation (6) can be split into the transformation at the geocenter (with $\mathbf{r}_E = 0$) and the position-dependent term:

$$T = T_{\text{geocenter}}(t) - \frac{1}{c^2} v^i_E r^i_E\,. \tag{16}$$

Computation of $T_{\text{geocenter}}(t)$ is discussed in the Section II B.

B. Transformation between TCB and TCG at the geocenter

The well-known differential relation between $T =$ TCG and $t =$ TCB at the geocenter reads

$$\frac{dT_{\text{geocenter}}}{dt} = 1 + F(t), \tag{17}$$

$$F(t) = -\frac{1}{c^2}\,\dot{A}(t) + \frac{1}{c^4}\,\dot{B}(t) + \mathcal{O}(c^{-5}). \tag{18}$$

This is the part of (6) for $r^i_E = 0$. Here $\dot{A}(t)$ and $\dot{B}(t)$ are defined by Eqs. (8)–(9). Again for the solar system considered as a system of N mass monopoles, one has

$$\dot{A}(t) = \frac{1}{2}\,v_E^2 + \sum_{A\neq E}\frac{GM_A}{r_{EA}}, \tag{19}$$

$$\begin{aligned}\dot{B}(t) = &-\frac{1}{8}\,v_E^4 + \left(\beta-\frac{1}{2}\right)\left(\sum_{A\neq E}\frac{GM_A}{r_{EA}}\right)^2 + (2\beta-1)\sum_{A\neq E}\left(\frac{GM_A}{r_{EA}}\sum_{B\neq A}\frac{GM_B}{r_{AB}}\right)\\ &+\sum_{A\neq E}\frac{GM_A}{r_{EA}}\Bigg(2(1+\gamma)v^i_A v^i_E - \left(\gamma+\frac{1}{2}\right)v_E^2 - (1+\gamma)v_A^2 + \frac{1}{2}a^i_A r^i_{EA}\\ &\qquad + \frac{1}{2}(v^i_A r^i_{EA}/r_{EA})^2\Bigg),\end{aligned} \tag{20}$$

where capital Latin subscripts A, B and C enumerate massive bodies, E corresponds to the Earth, M_A is the mass of body A, $\mathbf{r}_{EA} = \mathbf{x}_E - \mathbf{x}_A$, $r_{EA} = \lvert\mathbf{r}_{EA}\rvert$, $\mathbf{v}_A = \dot{\mathbf{x}}_A$, $\mathbf{a}_A = \dot{\mathbf{v}}_A$, a dot signifies time derivative with respect to $t =$ TCB, and $\mathbf{x}_A$ is the BCRS position of body A. The PPN parameters β and γ (both equal to 1 in general relativity) is given here for completeness, and normally should be put to 1 for practical calculations.

Standard way to calculate the relation (17) between TCB and TCG at the geocenter is to compute the integral defining this relation numerically. This way was used, for example, in [3] to compute the "time ephemeris" (see, Eqs. (2) and (24) of [3] or Eq. (3) of [5]). The integration is usually done using some advanced adaptive algorithm like Newton-Cotes one. The same way has been used in [13] in the similar situation to relate the proper time of *Gaia* with TCB.

In this note a different way (first suggested in [7]) is described. Namely, the differential equation relating TCB and TCG at the geocenter can be integrated numerically using any reasonable integrator for ordinary differential equations. The advantage of this approach is (1) a better control of numerical errors (any standard way to check the numerical accuracy, e.g. forth and back integration, can be used here), and (2) a simple exact way to invert the function (e.g., to compute not only TCG(TCB), but also TCB(TCG) at the geocenter).

Let us now define two functions $\Delta t(t)$ and $\Delta T(T)$ such that

$$T_{\mathrm{geocenter}} = t + \Delta t(t), \tag{21}$$

$$t = T_{\mathrm{geocenter}} - \Delta T(T_{\mathrm{geocenter}}). \tag{22}$$

Omitting subscript 'geocenter', one has two ordinary differential equations for $\Delta t(t)$ and $\Delta T(T)$:

$$\frac{d\Delta t}{dt} = F(t), \tag{23}$$

$$\frac{d\Delta T}{dT} = \frac{F(T - \Delta T(T))}{1 + F(T - \Delta T(T))}. \tag{24}$$

These relations are exact, while expression (18) for F is approximate. Initial conditions for these two differential equations are given by the IAU definitions of TCB and TCG: TCB $=$ TCG $= 32.184$ s on 1977, January 1, $0^h\ 0^m\ 0^s$ TAI at the geocenter. In terms of Julian Dates $JD_{\mathtt{TCB}}$ and $JD_{\mathtt{TCG}}$ in TCB and TCG, respectively one has:

$$\Delta t(JD_{\mathtt{TCB}} = 2443144.5003725) = 0, \tag{25}$$

$$\Delta T(JD_{\mathtt{TCG}} = 2443144.5003725) = 0. \tag{26}$$

Any reasonable integrator for ordinary differential equations can be used to integrate the differential equations (23)–(24) with given initial conditions (25)–(26). The accuracy of numerical integrations can be automatically checked, e.g., by integrating forth and back and comparing the results. The consistency of two independent integrations (one for $\Delta t(t)$ and another one for $\Delta T(T)$) can be cross-checked using identities $\Delta t(t) \equiv \Delta T(t + \Delta t(t))$ and $\Delta T(T) \equiv \Delta t(T - \Delta T(T))$.

C. Transformation between TDB and TT at the geocenter

Eqs. (17) and, correspondingly, (23)–(24) can be modified in an obvious way to relate any pair of the time scales TT and TDB at the geocenter. An efficient algorithm for that is described in [8]. Indeed, similarly to (21)–(22) one has

$$\mathtt{TT}_{\mathrm{geocenter}} = \mathtt{TDB} + \Delta\mathtt{TDB}(\mathtt{TDB}), \tag{27}$$

$$\mathtt{TDB} = \mathtt{TT}_{\mathrm{geocenter}} - \Delta\mathtt{TT}(\mathtt{TT}_{\mathrm{geocenter}}). \tag{28}$$

Omitting the subscript 'geocenter' one has

$$\frac{d\Delta\mathtt{TDB}}{d\mathtt{TDB}} = A_{\mathtt{TDB}} + B_{\mathtt{TDB}}\,\frac{d\Delta t}{dt}, \tag{29}$$

$$A_{\mathtt{TDB}} = \frac{L_B - L_G}{1 - L_B}, \tag{30}$$

$$B_{\mathtt{TDB}} = \frac{1 - L_G}{1 - L_B} = 1 + A_{\mathtt{TDB}}, \tag{31}$$

$$\frac{d\Delta\mathtt{TT}}{d\mathtt{TT}} = A_{\mathtt{TT}} + B_{\mathtt{TT}}\,\frac{d\Delta T}{dT}, \tag{32}$$

$$A_{\mathtt{TT}} = \frac{L_B - L_G}{1 - L_G}, \tag{33}$$

$$B_{\mathtt{TT}} = \frac{1 - L_B}{1 - L_G} = 1 - A_{\mathtt{TT}}, \tag{34}$$

where the derivatives $d\Delta t/dt$ and $d\Delta T/dT$ are defined by (23)–(24) and must be expressed as functions of TDB and TT, respectively, when used in (29) and (32). For TT the initial condition is the same as for TCG and TCB, while for TDB it is given by $\mathtt{TDB} = -6.55\times10^{-5}$ s for the same event. The initial conditions for $\Delta\mathtt{TDB}$ and $\Delta\mathtt{TT}$ are chosen according to the IAU 2006 Resolution defining TDB (see the definition of TDB in Section I above): for $JD_{\mathtt{TT}} = 2443144.5003725$ one has $JD_{\mathtt{TDB}} = 2443144.5003725 - 6.55\times10^{-5}/86400$ and vice versa. Therefore, one has

$$\Delta\mathtt{TDB}(JD_{\mathtt{TDB}} = 2443144.5003725 - 6.55\times10^{-5}/86400) = +6.55\times10^{-5}\ \text{s}. \tag{35}$$

$$\Delta\mathtt{TT}(JD_{\mathtt{TT}} = 2443144.5003725) = +6.55\times10^{-5}\ \text{s}, \tag{36}$$

Again (29) and (32) with initial conditions (35) and (36), respectively, can be numerically integrated.

Note that from (27)–(28) one gets the following identities that can be used to cross-check the results: $\Delta\mathtt{TT}(\mathtt{TT}) \equiv \Delta\mathtt{TDB}(\mathtt{TT} - \Delta\mathtt{TT}(\mathtt{TT}))$ and $\Delta\mathtt{TDB}(\mathtt{TDB}) \equiv \Delta\mathtt{TT}(\mathtt{TDB} + \Delta\mathtt{TDB}(\mathtt{TDB}))$.

III. TRANSFORMATION BETWEEN TG AND TCB

The relation between the proper time of *Gaia* $\tau = \mathtt{TG}$ and $t = \mathtt{TCB}$ is given by the basic relation of metric gravity theories:

$$\frac{d\tau}{dt} = 1 + f(t), \tag{37}$$

$$f = \frac{1}{c^2}\,\alpha(t) + \frac{1}{c^4}\,\beta(t) + \mathcal{O}(c^{-5}), \tag{38}$$

where $\alpha(t)$ and $\beta(t)$ are defined by the metric tensor of the BCRS :

$$\alpha = -\frac{1}{2}\,v_o^2 - \sum_A \frac{GM_A}{r_{oA}}, \tag{39}$$

$$\begin{aligned}\beta = &-\frac{1}{8}\,v_o^4 + \left(\beta - \frac{1}{2}\right)\left(\sum_A \frac{GM_A}{r_{oA}}\right)^2 + (2\beta - 1)\sum_A \left(\frac{GM_A}{r_{oA}} \sum_{B\neq A} \frac{GM_B}{r_{AB}}\right) \\ &+ \sum_A \frac{GM_A}{r_{oA}} \Bigg(2(1+\gamma)v_A^i v_o^i - \left(\gamma + \frac{1}{2}\right) v_o^2 - (1+\gamma)v_A^2 + \frac{1}{2}a_A^i r_{oA}^i \\ &\qquad + \frac{1}{2}(v_A^i r_{oA}^i / r_{oA})^2\Bigg). \end{aligned} \tag{40}$$

Here $\mathbf{r}_{oA} = \mathbf{x}_o - \mathbf{x}_A$, and $\mathbf{x}_o$ and $\mathbf{v}_o$ are the BCRS position and velocity of *Gaia*. Similar to (21)–(22) we introduce functions $\delta t(t)$ and $\delta\tau(\tau)$ as

$$\tau = t + \delta t(t), \tag{41}$$

$$t = \tau - \delta\tau(\tau). \tag{42}$$

Substituting these definitions in (37) one gets

$$\frac{d\delta t}{dt} = f(t), \tag{43}$$

$$\frac{d\delta\tau}{d\tau} = \frac{f(\tau - \delta\tau(\tau))}{1 + f(\tau - \delta\tau(\tau))}. \tag{44}$$

These are exact relations. The expression for $f(t)$ given above is approximate. The initial conditions for these differential equations come from the condition that $\tau =$ `TG` is numerically equal to $t =$ `TCB` at some moment $t = t_0$: $\tau(t_0) = \tau_0 = t_0$. This condition is equivalent to

$$\delta t(t_0) = 0, \tag{45}$$

$$\delta\tau(\tau_0) = 0. \tag{46}$$

The initial condition for `TG` is discussed in Section 3.5.1 of [1]. The actual value of t_0 was fixed after the launch (since the *Gaia* ephemeris should be available at this moment). The value of t_0 is available as Julian Date of `TCB` in *Gaia* Parameter Database as `GaiaParam.Mission.GAIAPROPERTIME_ZEROPOINT_TCB` and is equal to `JD2457023.5 TCB`.

IV. REPRESENTATION OF FUNCTIONS BY CHEBYSHEV POLYNOMIALS

A time ephemeris $y(t)$ (this can be any of the functions $\Delta t(t)$, $\Delta T(T)$, $\Delta TDB(TDB)$, $\Delta TT(TT)$, $\delta t(t)$ or $\delta\tau(\tau)$ or any other function of one argument) results from numerical integrations of the corresponding ordinary differential equations, t being the argument of the time ephemeris under consideration (not necessarily `TCB`). For time transformations one needs to evaluate $y(t)$ for any t within the validity interval $t_{\rm beg} \le t \le t_{\rm end}$. Depending on the used integrator one gets either values of $y(t)$ for fixed moments t_i or a sort of interpolating polynomials that can be evaluated for any t. One convenient method to construct a compact representation of $y(t)$ that can evaluate $y(t)$ for any t in the validity interval is to divide the whole validity interval into granules $[t_k, t_{k+1}]$ and represent $y(t)$ on each of the granules as a Chebyshev approximation polynomial:

$$y(x) \approx \widetilde{y}(x) = \sum_{i=0}^{n} a_i\, T_i(x), \tag{47}$$

where $T_i(x)$ are the Chebyshev polynomials of first kind, a_i are numerical coefficients, n is the order of the approximating polynomial, and $-1 \le x \le 1$ is a scaled independent variable (e.g. for interpolating between t_r and t_{r+1} one has $x = 2\,(t-t_r)/(t_{r+1}-t_r)-1$). In general, the size of the granule $\Delta t_r = t_{r+1} - t_r$ and the order n can be vary from granule to granule, but for our purposes it is sufficient to consider that $\Delta t_r = \Delta t = \text{const}$ and n is fixed. The Chebyshev polynomials $T_i(x)$ can be computed using the recurrence formula

$$T_0(x) = 1, \tag{48}$$

$$T_1(x) = x, \tag{49}$$

$$T_i(x) = 2\,x\,T_{i-1}(x) - T_{i-2}(x), \quad i \ge 2. \tag{50}$$

In the following, a single granule will be discussed. The construction of the approximating function should be done subsequently for all the granules.

The optimal values of coefficients a_i can be found in a number of well-known ways starting from a series of values of function $y(x)$ to be approximated. A convenient way to calculate a_i is described in [14]. The main difference to [14] is that we consider the approximating functions only for function $y(x)$ itself and not for its derivative. Although this is a technical detail, it is preferable to summarize all the formulas we need in the form that is directly used for *Gaia* time ephemerides.

Let us consider m points equally spaced in the granule

$$x_k = -1 + 2\,\frac{k-1}{m-1}, \quad k = 1, \dots, m \tag{51}$$

and assume that values $y_k = y(x_k)$ are known. Note that $x_1 = -1$ and $x_m = 1$ for any m. Then we have m equations with $n+1$ unknowns a_i:

$$\sum_{i=0}^{n} T_i(x_k)\, a_i = y_k. \tag{52}$$

This can be written in a matrix form

$$\mathbf{A}\,\mathbf{a} = \mathbf{y}, \tag{53}$$

$$\mathbf{A} = \begin{pmatrix} T_0(x_1) & T_1(x_1) & \dots & T_n(x_1) \\ T_0(x_2) & T_1(x_2) & \dots & T_n(x_2) \\ & & \dots & \\ T_0(x_m) & T_1(x_m) & \dots & T_n(x_m) \end{pmatrix}, \tag{54}$$

$$\mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{pmatrix}, \tag{55}$$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{pmatrix}. \tag{56}$$

In general, the more points (x_k, y_k) one has and the larger n one chooses, the better is the resulting approximation. Clearly, for $m > n + 1$ the system of equations should be solved in the sense of least squares:

$$\mathbf{a} = \left(\mathbf{A}^T\,\mathbf{A}\right)^{-1}\mathbf{A}^T\,\mathbf{y}, \tag{57}$$

where $\mathbf{A}^T$ is the transpose of $\mathbf{A}$. Note that the matrix $\mathbf{A}$ consists only of the values of Chebyshev polynomials $T_i(x_k)$, $i = 0, 1, \dots, n$ at points x_k, $k = 1, \dots, m$. Therefore, the matrix $\left(\mathbf{A}^T\,\mathbf{A}\right)^{-1}\mathbf{A}^T$ can be precomputed for any given n and m.

An important point is the behavior of the approximating function at the boundaries of the granules. If no additional conditions are added to (52) the approximating function is not continuous at the boundaries. This means that the polynomial (47) valid for $t_{k-1} \le t \le t_k$ and that valid for $t_k \le t \le t_{k+1}$ do not have the same value for $t = t_k$ (this is true for any k). This discontinuity can be eliminated if one requires than the approximating functions have exactly the values of $y(x)$ at the boundaries of the granules. Mathematically, this can be formulated as two constraints

$$\sum_{i=0}^{n} T_i(x_1)\,a_i = y_1, \tag{58}$$

$$\sum_{i=0}^{n} T_i(x_m)\,a_i = y_m, \tag{59}$$

that should be satisfied *exactly*. These constraints effectively eliminate two degrees of freedom of the approximating function (fix two coefficients among a_i). It is convenient to achieve this using Lagrange multipliers. The following system of equations should be solved (this system contains the system of normal equations for (53)):

$$\mathbf{B}\,\mathbf{p} = \mathbf{C}\,\mathbf{y}, \tag{60}$$

$$\mathbf{B} = \left(\begin{array}{cccc|cc} & & & & T_0(x_1) & T_0(x_m) \\ & \mathbf{A}^T\,\mathbf{A} & & & T_1(x_1) & T_1(x_m) \\ & & & & \vdots & \vdots \\ & & & & T_n(x_1) & T_n(x_m) \\ \hline T_0(x_1) & T_1(x_1) & \ldots & T_n(x_1) & 0 & 0 \\ T_0(x_m) & T_1(x_m) & \ldots & T_n(x_m) & 0 & 0 \end{array}\right), \tag{61}$$

$$\mathbf{C} = \left(\begin{array}{c} \mathbf{A}^T \\ \hline 1\,0\,0 \ldots 0\,0\,0 \\ 0\,0\,0 \ldots 0\,0\,1 \end{array}\right), \tag{62}$$

$$\mathbf{p} = \left(\begin{array}{c} a_0 \\ a_1 \\ \ldots \\ a_n \\ \hline \lambda_1 \\ \lambda_2 \end{array}\right), \tag{63}$$

where λ_i are the Lagrange multipliers, the values of which are of no interest in the context of this note. The coefficients a_i can be computed from the first $n+1$ rows of matrix $\mathbf{B}^{-1}\mathbf{C}$:

$$\mathbf{p} = \mathbf{B}^{-1}\,\mathbf{C}\,\mathbf{y}. \tag{64}$$

Again this can be computed independently of y_k and the granule size Δt.

In principle, an approximation to the derivative $dy(x)/dx$ can be easily computed from (47) using the fact that

$$\frac{d}{dx}\,T_i(x) = i\,U_{i-1}(x), \tag{65}$$

where $U_i(x)$ are the Chebyshev polynomials of the second kind

$$U_0(x) = 1, \tag{66}$$

$$U_1(x) = 2\,x, \tag{67}$$

$$U_i(x) = 2\,x\,U_{i-1}(x) - U_{i-2}(x), \quad i \geq 2. \tag{68}$$

Note that using constraints (58)–(59) one gets the approximating function that is continuous, but not differentiable at the boundaries: the derivative of (47) is not necessary continuous at the boundaries of the granules. If the exact values of the derivatives $y'(x_1) = y'_1$ and $y'(x_m) = y'_m$ at the boundaries are available, they can be used to formulate two more constraints for the approximating function

$$\sum_{i=1}^{n} i\,U_{i-1}(x_1)\,a_i = y'_1, \tag{69}$$

$$\sum_{i=1}^{n} i\,U_{i-1}(x_m)\,a_i = y'_m, \tag{70}$$

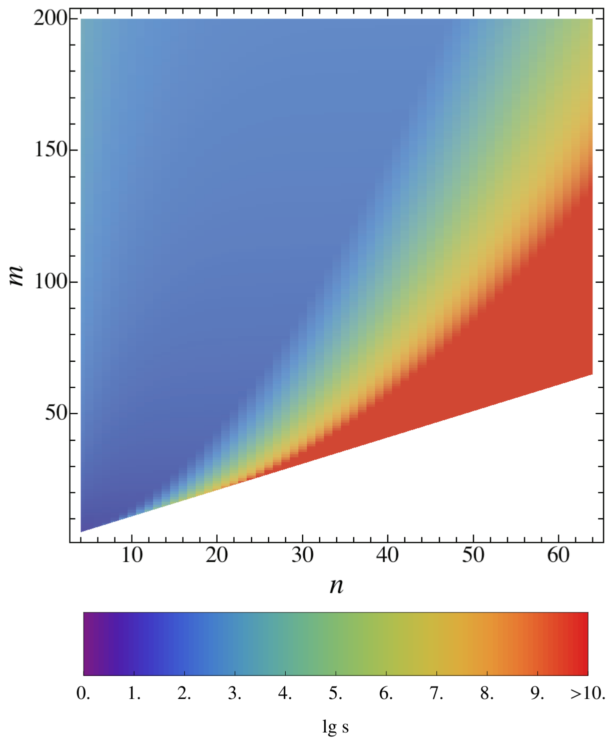

FIG. 1: The values of the condition number s of $\mathbf{B}$ as function of the order of approximating function n and the number of points m. The condition number is computed using singular value decomposition (SVD) as ratio of the largest and smallest singular values. All values greater or equal to 10^{10} are shown red.

that can also be implemented using Lagrange multipliers. One could add further constraints to get also higher derivatives continuous at the boundaries, but we will not consider this explicitly. In case of time ephemerides it is not expected that they will be used to compute the derivatives. Therefore, it seems to be sufficient to include only two constraints (58)–(59) (this guarantees that the approximating function is continuous, not necessarily differentiable at the boundaries of the granules).

The coefficients a_i can be computed a linear combination of a set of m values of y_k:

$$a_i = \sum_{k=1}^{m} \alpha_k^i \, y_k, \qquad i = 0, \dots n, \tag{71}$$

where α_k^i are the corresponding elements of matrix $\mathbf{B}^{-1}\mathbf{C}$ (or matrix $\left(\mathbf{A}^T\,\mathbf{A}\right)^{-1}\mathbf{A}^T$ if no constraints are considered) as given by (64) (or (57) without constraints). Note that coefficients α_k^i depend on n and m, but not on the size of the granules Δt.

In order to compute α_k^i one has to invert matrix $\mathbf{B}$ (or $\mathbf{A}^T\,\mathbf{A}$). Figure 1 shows the condition number of matrix $\mathbf{B}$. It is clear that for larger n ($n > 20$) and m only slightly larger than $n+1$ the matrix is (very) close to degeneracy (for example, for $n = 64$ and $m = n+1 = 65$ the condition number of $\mathbf{B}$ is 1.13×10^{33}). However, if m is chosen sufficiently large the condition number remains moderate even for large n. Interestingly, one can find that the minimal condition number for given n is reached for $m = m_{\rm min}$, where

$$m_{\min} \approx \left[\, 0.71\,(1+n) + 0.075 n^2 \,\right] + 1, \tag{72}$$

$[x]$ being the integer part of x. With $m = m_{\min}$, the condition number of matrix $\mathbf{B}$ remains below 10^3 for $n \le 100$. Moreover, Figure 1 shows that the condition number grows only moderately with m for $m > m_{\min}$. Direct numerical computations show that the condition number of $\mathbf{A}^T\,\mathbf{A}$ does not exceed that of $\mathbf{B}$ for $4 \le n \le 64$ and $n+1 \le m \le 100$.

In order to avoid numerical stability problems the computation of α^i_k has been first implemented using arbitrary-precision arithmetic using *Mathematica*. A library of the values α^i_k for $4 \le n \le 32$ and $n+1 \le m \le 120$ with the constraints (58)–(59) has been pre-computed using arbitrary-precision arithmetic and can be used directly in (71). This *Mathematica* software to calculate α^i_k for arbitrary n and m and for various sets of constraints at the boundaries is available from the author. On the other side, using $m \ge m_{\min}$ it is safe to compute α^i_k directly in the [Java-]code using standard 64-bit arithmetic. Precomputed values of α^i_k make the conversion to Chebyshev representation (47) very fast as soon as n, m, and the granule size Δt are fixed.

Choice of n, m and the size of granules Δt is a non-trivial procedure which requires direct trials. Reducing the granule size Δt and increasing n both lead to more accurate approximating functions. Smaller size of the granules lead to larger data volume necessary to represent $y(t)$ for the whole validity interval $t_{\rm beg} \le t \le t_{\rm end}$. Generally speaking, the faster $y(x)$ is changing, the larger n and/or the smaller granule size Δt are required to approximate it with a given accuracy. For a given $y(x)$ higher values of n require more arithmetic operations to compute the approximating function. For a given approximation accuracy the choice of n, m and Δt is, therefore, a typical optimization problem between computational efforts and data volume. For the time transformations in *Gaia* it seems reasonable to keep the evaluation effort as low as possible while the amount of data seem to play a secondary role. Our numerical experiments with time ephemerides show that in order to attain the accuracy of 0.01 ns for the granule size of 1 day it is sufficient to have $n = 4$. This means that the time ephemerides can be evaluated at a cost of a few multiplications and additions.

Once the approximating function is computed it is easy to check its accuracy by comparing its values with $y(x)$ at some test values x_j which do not necessary coincide with the points used in (53) to construct the approximating function. If the accuracy check shows deviations to $y(x)$ larger than some desired level, the parameters of the approximating function should be changed and the process iterated. The simple strategy chosen for *Gaia* time ephemerides is described below in Section V C.

The most efficient way to evaluate a linear combination of Chebyshev polynomials is given by the Clenshaw algorithm described, e.g., by Eq. (5.8.11) of [15]. The Clenshaw algorithm for (47) can be written as:

```
$z = 2\,x$
$d_{n+1} = d_{n+2} = 0$
for $k = n,\ n-1,\ \dots,\ 1$
    $d_k = z\,d_{k+1} - d_{k+2} + a_k$
next $k$
$f(x) = x\,d_1 - d_2 + a_0$
```

The Clenshaw algorithm is numerically stable and superior to the straightforward recurrent computation of the values of the Chebyshev polynomials themselves. Note that this algorithm with a slight modification can also be used to compute the derivative of (47) using (65)–(68).

V. JAVA IMPLEMENTATION

A. User-oriented time transformations: class `BasicTimeTransformations`

The GaiaTools package `gaia.cu1.tools.time` contains most of the functionality required by the end users in *Gaia* DPAC. So,me parts of the code described below is intended for internal use in `gaia.cu1.tools.time` and by the software that creates the transformations between `OBMT` and `TG` (this software and the algorithms used by that software are not part of this document).

The time transformations described above are all implemented in class `BasicTimeTransformations`. The time transformations can be split into three groups: (1) location-independent, (2) with assumed location (*Gaia* or the geocenter), (3) location-dependent. The time transformation methods of the first two kinds have the signature

```
long XXX2YYYccc(long ns, double refEpoch)
```

These methods implement time transformation from time scale `XXX` to scale `YYY`. The input time moment is given by two input parameters: `ns` gives the offset in nanoseconds from `refEpoch`, that represents a Julian date in the input time scale $JD_{\mathtt{XXX}} = \mathtt{refEpoch}$. The output time moment is given by the return value of the function being the offset in nanoseconds from the epoch $JD_{\mathtt{YYY}} = \mathtt{refEpoch}$. Note that `refEpoch` is used as value of the epoch in both `XXX` and `YYY`. The last part `ccc` of the name is empty in case of location-independent transformations and indicates the location if a location was assumed (it could be `AtGeocenter` or `AtGaia`). The location-dependent transformations have one additional parameter:

```
long XXX2YYY(long ns, double refEpoch, GVector3d p)
```

where `p` is the vector representing the coordinate position of the event, at which the time transformation should be evaluated. Depending on the transformation it can be vector $\mathbf{r}_E = \mathbf{x} - \mathbf{x}_E$ (the difference of the BCRS position $\mathbf{x}$, at which the transformation should be computed, and the BCRS position $\mathbf{x}_E$ of the Earth) or the GCRS coordinates $\mathbf{X}$ of the event. Exact meaning of `p` is always described in the comments of the corresponding method (Javadoc).

The implemented transformations are:

– between `UTC` and `TAI` (location-independent): `UTC2TAI`, `TAI2UTC`

– between `TT` and `TAI` (location-independent): `TT2TAI`, `TAI2TT`

– between `GPS` and `TAI` (location-independent): `GPS2TAI`, `TAI2GPS`

– between `TT` and `TCG` (location-independent): `TT2TCG`, `TCG2TT`

- between TDB and TCB (location-independent): TDB2TCB, TCB2TDB
- between TDB and TT at the geocenter (assumed location): TDB2TTAtGeocenter, TT2TDBAtGeocenter
- three version of the transformations between TCB and TCG:
  - at the geocenter (assumed location): TCB2TCGAtGeocenter, TCG2TCBAtGeocenter
  - at the location of *Gaia* (assumed location): TCB2TCGAtGaia, TCG2TCBGAtGaia
  - at a given spatial location (location-dependent): TCB2TCG, TCG2TCB
- between TG and TCB (location-independent): TG2TCB, TCB2TG

All these time transformation methods are static and can be used without instantiation of BasicTimeTransformations. All necessary initializations (internal constants, time ephemerides, Earth and *Gaia* ephemerides, etc.) are done in a static block executed while loading the class by the Java machine (JVM).

B. Evaluating pre-computed time ephemerides: class TimeEphemeris

The use of time ephemerides consists of two steps: (1) the creation (computation) of time ephemeris and (2) evaluation of the time ephemerides. The creation of time ephemerides is discussed in Section V C. The evaluation of pre-computed time ephemerides is implemented in class TimeEphemeris. The class can deal with the following types of time ephemerides (as defined by the enumeration TimeEphemerisType):

- TCGminusTCBparametrizedByTCB:
  represents $\Delta\mathtt{TCB}(\mathtt{TCB})$ from $\mathtt{TCG} = \mathtt{TCB} + \Delta\mathtt{TCB}(\mathtt{TCB})$ at the geocenter ($\delta t(t)$ in (21), where $t = \mathtt{TCB}$ and $T = \mathtt{TCG}$);
- TCBminusTCGparametrizedByTCG:
  represents $\Delta\mathtt{TCG}(\mathtt{TCG})$ from $\mathtt{TCB} = \mathtt{TCG} - \Delta\mathtt{TCG}(\mathtt{TCG})$ at the geocenter ($\Delta T(T)$ in (22), where $t = \mathtt{TCB}$ and $T = \mathtt{TCG}$);
- TTminusTDBparametrizedByTDB:
  represents $\Delta\mathtt{TDB}(\mathtt{TDB})$ from $\mathtt{TT} = \mathtt{TDB} + \Delta\mathtt{TDB}(\mathtt{TDB})$ at the geocenter ($\Delta\mathtt{TDB}(\mathtt{TDB})$ in (27));
- TDBminusTTparametrizedByTT:
  represents $\Delta\mathtt{TT}(\mathtt{TT})$ from $\mathtt{TDB} = \mathtt{TT} - \Delta\mathtt{TT}(\mathtt{TT})$ at the geocenter ($\Delta\mathtt{TT}(\mathtt{TT})$ in (28));
- TGminusTCBparametrizedByTCB:
  represents $\delta\mathtt{TCB}(\mathtt{TCB})$ from $\mathtt{TG} = \mathtt{TCB} + \delta\mathtt{TCB}(\mathtt{TCB})$ ($\delta t(t)$ in (41), where $t = \mathtt{TCB}$ and $\tau = \mathtt{TG}$);
- TCBminusTGparametrizedByTG:
  represents $\delta\mathtt{TG}(\mathtt{TG})$ from $\mathtt{TCB} = \mathtt{TG} - \delta\mathtt{TG}(\mathtt{TG})$ ($\delta\tau(\tau)$ in (42), where $t = \mathtt{TCB}$ and $\tau = \mathtt{TG}$).

Two representations of time ephemerides are supported (as defined by the enumeration `TimeEphemerisRepresentation`):

– `CHEBYSHEV`: piecewise Chebyshev polynomials (see Section IV);

– `SPLINE`: cubic splines with uniformly distributed knots (splines are fully implemented, but not used in the normal work and not described here).

We note here that for the *Gaia* data processing only piecewise Chebyshev polynomials are used. Class `TimeEphemeris` has two constructors. The first one is intended for the user. The only parameter of this constructor of `TimeEphemeris` is the type of the time ephemeris (again as defined by the enumeration `TimeEphemerisType`). The constructor loads the requested time ephemeris and prepares the efficient evaluation. The other constructor allows to create an instance of `TimeEphemeris` specifying all the internal data of a time ephemeris as parameters. This constructor is used in package `createTimeEphemerides` (Section V C) to check the validity of newly created time ephemerides.

The object of class `TimeEphemeris` can be interrogated for various constants concerning the time ephemeris: its type, its validity interval, its representation, and further details of representation. Finally, method

```
double getValue(long ns, double refEpoch)
```

allows one to evaluate the time ephemeris for a given moment of the input time scale. The format of the input moment has the same form as described in Section V A: offset `ns` in nanoseconds from a reference epoch `refEpoch` in Julian days. We note that the default value of `refEpoch` is given in the *Gaia* Parameter Database as Julian Date `GaiaParam.Nature.JULIANDATE_J2010` and is equal to `2455197.5`. As it is already mentioned above, these value is used for Julian Dates in different time scales for different time ephemerides and thus don't correspond to the same physical moment of time. The output value is the value of the corresponding time ephemeris in seconds.

We note that we have four more time ephemeris types in `TimeEphemerisType`:

`TGminusOBMTparametrizedByOBMT`,

`OBMTminusTGparametrizedByTG`,

`TCBminusOBMTparametrizedByOBMT`,

`OBMTminusTCBparametrizedByTCB`.

These types of the time ephemerides are used by HATT [9, 10] to relate the *Gaia* clock readings `OBMT` with `TG` and directly with `TCB`. The semantics of these ephemerides is clear from the names and the analogous time ephemerides discussed above.

C. Creation of the time ephemerides: package `createTimeEphemerides`

The formulas given in Sections II B and II C can be directly used to calculate the relations between `TCB` and `TCG` or `TDB` and `TT` at the geocenter for any given ephemeris providing the masses of gravitating bodies, their position, velocities and accelerations. The authors of the solar system ephemerides can compute these relations with better accuracy during the process of construction of the ephemeris (compared to the accuracy that can be achieved when constructing the time ephemeris a posteriori with an export version of solar system ephemeris). The accuracy is better since the authors of solar system ephemerides have information on the additional bodies (asteroids, etc.) used in the dynamical model and not included into the export versions of the ephemerides. One more advantage of integrating the relations between `TCB` and `TCG` at the geocenter during the construction of ephemerides is that the export version of solar system ephemerides are less accurate by themselves compared to the raw numerical integrations, although this second argument plays only a secondary role.

For practical purposes one needs only one pair of transformation: either between `TCB` and `TCG` or between `TDB` and `TT`. The other transformation can be computed from the implemented one using the definition of `TDB` as function of `TCB` and that of `TT` as function of `TCG` (see Section I above). In the standard implementation the relation between `TCB` and `TCG` is computed via the relation between `TDB` and `TT` at the geocenter. Both transformations from `TCB` to `TCG` and from `TDB` to `TT` are parts of the export versions of INPOP ephemerides. The export format of INPOP is converted into internal format suitable for the DPAC Java environment by class `InpopTimeEphemerisImporter`.

Since the solar system ephemeris in GaiaTools does not normally cover $JD = 2443144.5003725$ (1 January 1977), it is not possible to use the correct initial conditions (35)–(36). This means that within the *Gaia* Java environment it is normally not possible to compute the relations between `TCB` (or `TDB`) and `TCG` (or `TT`) at the geocenter. As clarified above, this is also not required since the time ephemeris between `TT` and `TDB` (and between `TCG` and `TCB`) is included in the INPOP export ephemerides and can be used directly.

The code used for the transformation between `TG` and `TCB` (see Section III) is almost identical to the code that can compute the relations between `TCB` (or `TDB`) and `TCG` (or `TT`) at the geocenter. As a consistency check of the code, the used computational method, and the INPOP time ephemeris, the relation between `TT` and `TDB` at the geocenter is computed by class `TtTdbIntegrator`. The numerical integration for (29)–(32) is performed using artificial initial conditions

$$\Delta\mathtt{TDB}(JD_{\mathtt{TDB}} = JD_0) = 0. \tag{73}$$
$$\Delta\mathtt{TT}(JD_{\mathtt{TT}} = JD_0) = 0, \tag{74}$$

where JD_0 is an artificially chosen moment of time lying within the validity period of the solar system ephemeris in GaiaTools. As a result one gets functions Δ`TDB` and Δ`TT` that differ from the correct ones by an additive constant. Then the difference between Δ`TDB` from the INPOP and our numerical integration is computed and the constant part of the difference is subtracted. The resulting difference can be analyzed as usual. Our calculations have demonstrated that the INPOP time ephemeris is restored by our direct numerical computation to about 10^{-17} in rate and 50 ps for quasi-periodic terms. This accuracy is fully sufficient for *Gaia*. Nevertheless, the reasons of this difference are also clear. First, the INPOP time ephemeris is computed using asteroids in (19)–(20); the asteroids are not used in our integration since they are not included into the export version of the INPOP ephemerides. Second, we use the export version of the INPOP ephemeris while the INPOP time ephemeris is computed with slightly different version of INPOP coming directly from numerical integrations. Other reasons may exist [12].

Class `TtTdbIntegrator` with flag `testOnly` set to `true`, computes functions Δ`TDB` and

Δ`TT` with initial conditions (73)–(74), checks the consistency of numerical integrations and interpolations, and compares the results for ΔTDB with the INPOP time ephemeris. If flag `testOnly` is set to `false`, class `TtTdbIntegrator` computes functions Δ`TDB` and Δ`TT`, checks the consistency of numerical integrations and interpolations, represents the resulting function in terms of cubic splines or Chebyshev polynomials (see Section IV) and stores the resulting representations in a form which can be used by the class `TimeEphemeris` to evaluate the time ephemerides for any suitable value of argument. In this case the corresponding time transformation can be computed without the INPOP time ephemeris. This possibility is not supposed to be used in normal case.

Class `TcgTcbIntegrator` computes functions Δt and ΔT defined in (21)–(22), checks the consistency of numerical integrations and interpolations, represents the resulting function in terms of cubic splines or Chebyshev polynomials (see Section IV) and stores the resulting representations in a form which can be used by the class `TimeEphemeris` to evaluate the functions for any suitable value of argument. Using this class the corresponding time transformation (that between `TCB` and `TCG` at the geocenter) can be computed without the INPOP time ephemeris. This possibility is not supposed to be used in normal case since the transformation between `TCB` and `TCG` at the geocenter is computed from that between `TDB` and `TT` at the geocenter, and the latter is taken from the INPOP ephemeris.

Class `TgTcbIntegrator` computes functions δt and δT defined in (41)–(42), checks the consistency of numerical integrations and interpolations, represents the resulting function in terms of cubic splines or Chebyshev polynomials (see Section IV) and stores the resulting representations in a form which can be used by the class `TimeEphemeris` to evaluate the functions for any suitable moment of time. This class represents the normal source of the relation between `TG` and `TCB` and is intended to be used each time new version of the *Gaia* ephemeris is delivered to DPAC.

Numerical integrations are performed using the ODEX (Gragg-Bulirsch-Stoer) integrator as implemented in `apache.common.math`. For each integration the resulting numerical accuracy is automatically checked by integrating forth and back. The consistency of each pair of functions $\big(\Delta t(t), \Delta T(T)\big)$, $\big(\Delta\mathtt{TDB}(\mathtt{TDB}), \Delta\mathtt{TT}(\mathtt{TT})\big)$, and $\big(\delta t(t), \delta\tau(\tau)\big)$ is automatically checked using the identities like $\Delta t(t) = \Delta T(t + \Delta t(t))$ and $\Delta t(T - \Delta T(T)) = \Delta T(T)$ (see above). Class `NumericalIntegration` is a convenient driver for ODEX. It implements integrations forth and back for the accuracy check, provides access to the statistics, etc. Class `TimeDerivative` implements all six functions on the right-hand sides of the differential equations for the time ephemerides.

Class `CoefficientsCreator` contains one static method `GenerateCoefficients` that allows one to compute the coefficients α_k^i needed to compute Chebyshev approximating functions. Coefficients α_k^i can be computed for any n and m and in two versions: unconstrained and constrained in such a way that the approximating function is continuous at the edges of the granules (see Section IV).

Class `Util` contains a number of static methods. Many of the methods are only used for internal purposes and will not be mentioned here. Method `createChebyshev` allows one to create the Chebyshev representation of a function. Section IV discusses this process. For *Gaia* time ephemeris a simple procedure to choose order of polynomials n, number of points per granule m and the size of granule Δt is adopted. Two parameters are fixed: $\Delta t = 1$ day, $m = 49$. First, coefficients α_k^i are calculated for $n = 4$ and $m = 49$ (the condition number of the corresponding matrix is 168.8). Second, the approximating function is constructed with these α_k^i and $\Delta t = 1$ day. Third, the accuracy of the approximating function is checked on a grid with the time step of 30 sec. If the error exceeds 0.01 ns the coefficients α_k^i for $n = 5, 6, \ldots, 21$ and $m = 49$ (the condition number is always less than 168.8) are subsequently calculated and used to construct the approximating function with $\Delta t = 1$ day. If the accuracy of 0.01 ns could not be reached even with $n = 21$, the code gives up suggesting to check the situation manually. In this case, the function to be approximated has some unusual peculiarities (e.g., discontinuities, etc.). Our tests show that normally $n = 4$ is fully sufficient to achieve the accuracy of 0.01 ns. This makes the calculation of time ephemerides rather cheap.

Method `createSpline` creates, for a given function, a cubic spline with evenly distributed knots. In principle, this is an alternative for the Chebyshev representation. It is also fully implemented and can be used for all time ephemerides. However, detailed tests have shown that the splines is worse than the Chebyshev polynomials not only in the number of required coefficients (this is obvious and was expected), but also in the time needed to evaluate them. Therefore, the splines are not normally used.

Method `createFits` creates a fits file (see Section V D) with binary table fully defining a time ephemeris. These tables can be read by the standard constructor of `TimeEphemeris`.

Method `createTimeEphemeris` is a convenient driver creating the requested time ephemeris (e.g., as fits file) from a continuous representation of the function resulted from numerical integration. The method creates a standard representation of the time ephemeris (spline or Chebyshev polynomials), checks the accuracy of the time ephemeris and creates corresponding fits file. This method uses `CoefficientsCreator.GenerateCoefficients`, `createChebyshev` or `createSpline`, and `createFits`.

D. Integration into GaiaTools

Two classes `BasicTimeTransformations` and `TimeEphemeris` described above have been integrated into GaiaTools by Uwe Lammers and are in package `gaia.cu1.tools.time`. Package `createTimeEphemerides` remains outside of GaiaTools as `gaia.cu3.remat.time.createTimeEphemerides`. The main change of the code concerns one more format for the time ephemerides: in addition to the fits files one can create and use gbin files (a database format used in GaiaTools). To this end, both `TimeEphemeris` and classes of `createTimeEphemerides` have got additional methods which are used internally. The details of the configuration needed to use the functionality of the *Gaia* time ephemerides is described in the corresponding technical documentation.

[1] Bastian, U., 2007, Reference Systems, Conventions and Notations for *Gaia*, GAIA-CA-SP-ARI-BAS-003-06

[2] Bastian U., 2010, OBMT, the technical time coordinate of DPAC. GAIA-C1-TN-ARI-BAS-030-03

[3] Fukushima T. 1995, Time Ephemeris, Astron. Astrophys., 294, 895

[4] Harada W., Fukushima T. 2003, Harmonic decomposition of time ephemeris TE405, Astron.J., 126, 2557

[5] Irwin A.W., Fukushima T. 1999, A numerical time ephemeris of the Earth, Astron. Astrophys., 348, 642

[6] Fairhead L., Bretagnon P. 1990, An analytical formula for the time transformation TB-TT, Astron. Astrophys., 229, 240

[7] Klioner S.A. 2008, Relativistic astrometry and astrometric relativity, in: A Giant Step: from Milli- to Micro-arcsecond Astrometry, W.Jin, I.Platais, M.Perryman (eds.) Proc. of the IAU Symposium 248, Cambridge University Press, Cambridge, 356

[8] Klioner S.A., Gerlach E., Soffel M.H. 2010, Relativistic aspects of rotational motion of celestial bodies, in: Relativity in Fundamental Astronomy: Dynamics, Reference Frames and Data Analysis, S.A. Klioner, P.K. Seidelmann, M.H. Soffel (eds.), Proc. of the IAU Symposium 261, Cambridge University Press, Cambridge, 112

[9] Klioner, S. 2015, High-accuracy timing for *Gaia* data from one-way time synchronization, in: Recent Developments and Prospects in Ground-Based and Space Astrometry, ed. by N. Capitaine, Z. Malkin (Paris Observatory, Paris, 2015), p. 55

[10] Klioner S.A., Geyer R., Steidelmüller H., Butkevich A.G. 2017, Data Timing, Time Transfer and On-board Clock Monitoring for Space Astrometry with Gaia, Space Science Reports, 212, 1423DOI: 10.1007/s11214-017-0419-8

[11] Lammers U. 2010, *private communication*

[12] Manche H. 2010, *private communication*

[13] Mignard F., Crosta M.T., Klioner, S.A. 2004, Relation between the *Gaia* proper time and TCB. GAIA-FM-020

[14] Newhall X.X. 1989, Numerical representation of planetary ephemerides, Celestial Mechanics, 45, 305

[15] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P., 1992, Numerical Recipes In Fortran: the art of scientific computing, second edition, Cambridge University Press, New York

[16] Soffel M., Klioner S.A., Petit G., Wolf P., Kopeikin S.M., *et al.* 2003, The IAU 2000 Resolutions for astrometry, celestial mechanics, and metrology in the relativistic framework: explanatory supplement, Astron.J., 126, 2687