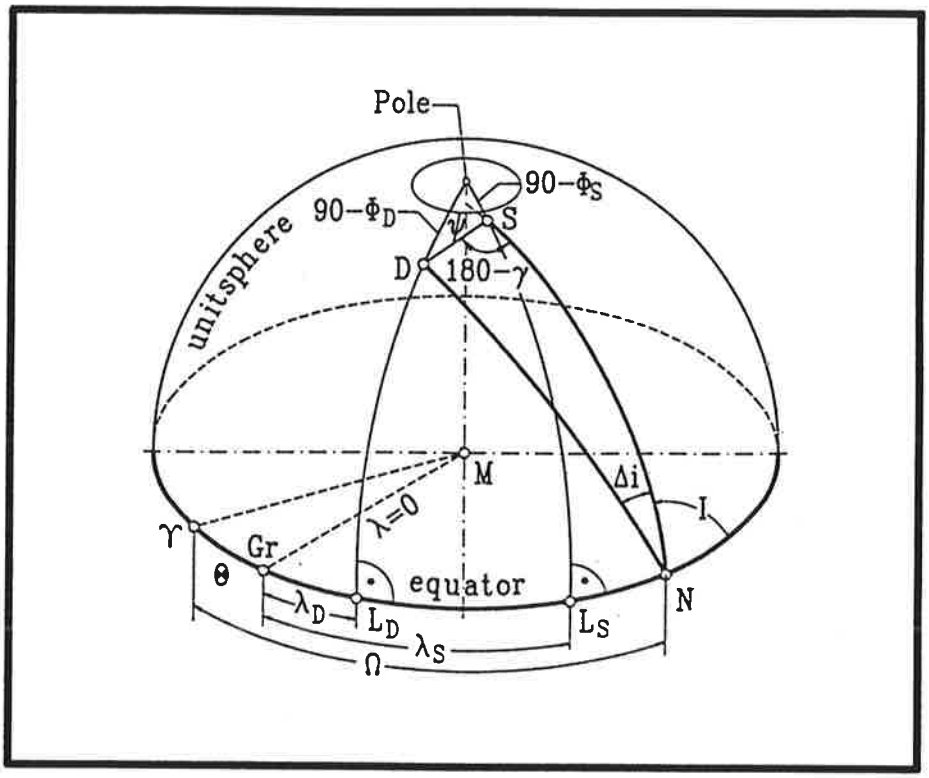
Max-Planck-Institut
für Meteorologie

REPORT No.75



ON THE NAVIGATION OF POLAR ORBITING METEOROLOGICAL SATELLITES

by

JAROSLAV KLOKOCNIK • PETER SCHLÜSSEL
JAN KOSTELECKY • HARTMUT GRASSL

HAMBURG, OCTOBER 1991

ON THE NAVIGATION OF POLAR ORBITING METEOROLOGICAL SATELLITES

J. Klokocník [1,3], P. Schlüssel [2], J. Kostelecky [1,4], H. Grassl [1,2]

1. Max-Planck-Institut für Meteorologie, Hamburg, Germany
2. Meteorologisches Institut, Universität Hamburg, Germany
3. permanent address: Astronomical Institute, Czech Academy Sci. CS-25165 Ondrejov Observatory
4. permanent address: Research Institute for Geodesy, Prague CS-25066 Zdiby 98

ISSN 0937–1060

AUTHORS:

JAROSLAV KLOKOCNIK [3]

JAN KOSTELECKY [4]

HARTMUT GRASSL [2]

MAX–PLANCK–INSTITUT
FÜR METEOROLOGIE

PETER SCHLÜSSEL

2 METEOROLOGISCHES INSTITUT
UNIVERSITÄT HAMBURG
BUNDESSTRASSE 55
D– 2000 HAMBURG 13
GERMANY

permanent address:

3 ASTRONOMICAL INSTITUTE
CZECH ACADEMY SCI
CS– 25165 ONDREJOV OBSERVATORY

4 RESEARCH INSTITUTE
FOR GEODESY
PRAGUE
CS– 25066 ZDIBY 98

MAX–PLANCK–INSTITUT
FÜR METEOROLOGIE
BUNDESSTRASSE 55
D–2000 HAMBURG 13
F.R. GERMANY

Tel.: +49 (40) 4 11 73–0
Telex: 211092 mpime d
Telemail: MPI.METEOROLOGY
Telefax: +49 (40) 4 11 73–298

1. *Introduction*

Satellite navigation is a process of identifying the space and time coordinates of the satellite data products (e.g. images). The satellite navigation without ground control points is tied to satellite orbit determination and prediction (ephemerides), as shown in Fig. 1.

Satellite observations of various kinds and accuracies are gathered from a global network of observing stations by a space agency. To compute the orbits, the observations are merged together, weighted according to their quality, and used in a least-squares adjustment - orbit determination process - to find or improve the orbital elements and various physical parameters (like the Earth gravity field and atmospheric parameters, geocentric coordinates of the tracking stations, etc.). These values are used for orbit prediction and ephemerides that enable further observations from the network for a next orbit determination. Due to the perturbing forces acting on the satellite like the Earth gravity variations, luni-solar attraction, atmospheric drag and solar pressure, the orbit differs from an idealized Keplerian ellipse, and evolves in time. Thus, the orbital elements cannot be determined once for all. On the contrary, a set of "new" observations obtained during a few days around date t_1 will be used to update the elements from the previous epoch t_0. This is an endless process.

To perform the observations, the tracking stations need to know when and where on the sky the satellites will fly over the individual stations. This information is known as ephemeris data. "General" ephemerides inform on the satellite's pass, on Sun's illumination and on conditions of simultaneous observability of the satellite from more stations. "Detailed" ephemeris data are used to control the mount of the tracking instument (camera, laser radar, etc.) that follows the satellite's motion on the sky.

Navigation of meteorological satellites can, therefore, be understood as an application of orbit determination and prediction, whereby the accuracy requirements are moderate in comparison with the strict demands by geodesy and geodynamics. The required observations may be of lower accuracy (radar, radio) and the force (perturbation) model to account for the orbit perturbations during the orbit adjustment is simplified.

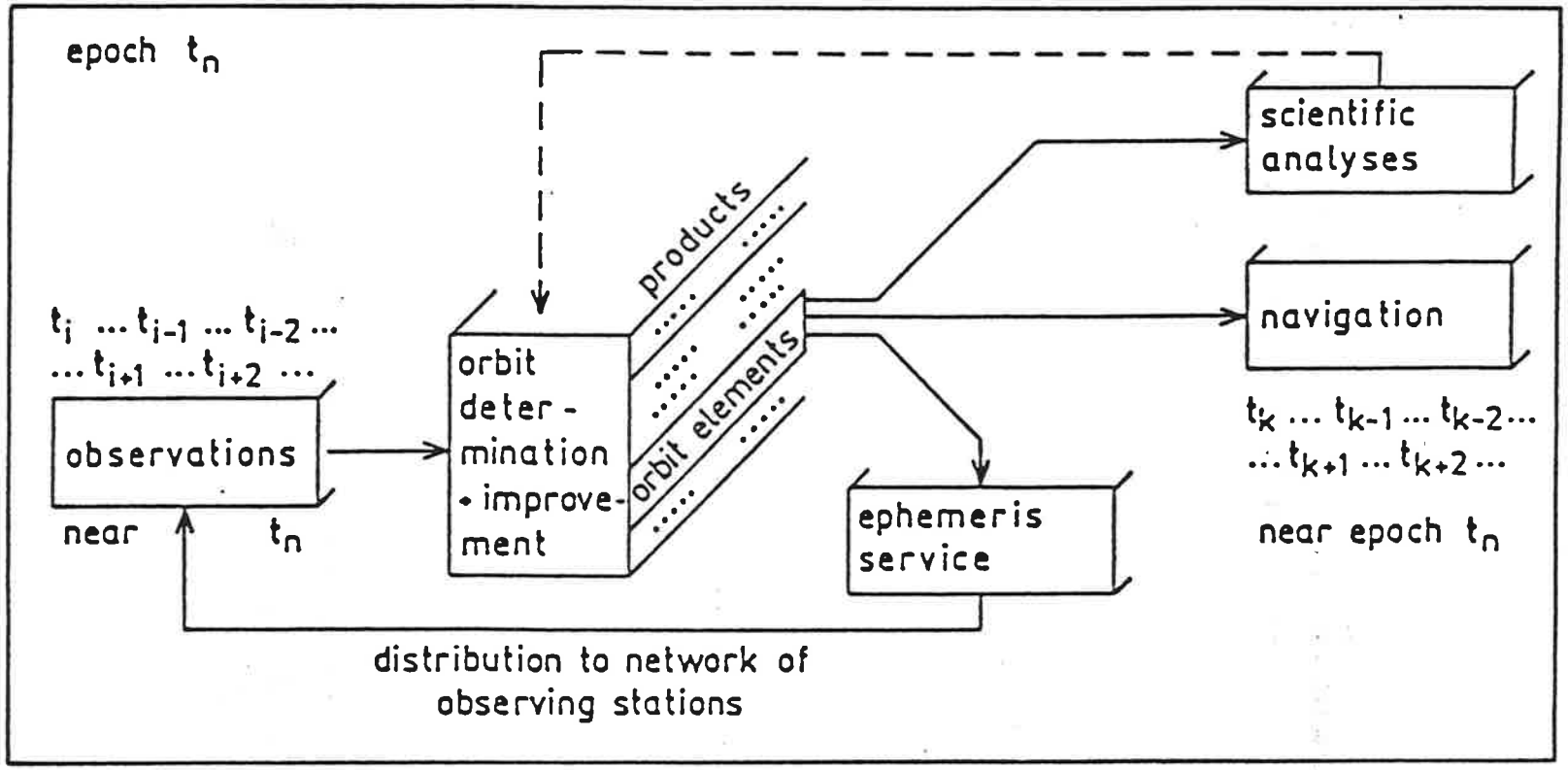


Fig. 1: Information flow between satellite observations, orbit determination, ephemeris data and navigation. Reproduced from Straka et al. (1991).

In satellite navigation, Ho and Asem (1986) as well as Emery at al. (1989) distinguish direct and inverse referencing. The former is to locate a subsatellite point or a pixel of an image by assigning a scan line l and a pixel number p for a given time from given orbital elements, i.e. to find geographic or geodetic coordinates of that point. Inverse referencing means that the geographic coordinates of a place of interest are given and we seek the corresponding image coordinates (l, p), utilizing given orbital elements.

From the viewpoint of satellite dynamics, mean orbital elements in epoch t_0 are given and we compute the satellite position in t ; the time elapsed since t_0 is not too large. Basically this task consists of a transformation between mean elements in t_0 and osculating elements in t as well as of a transformation from coordinates not connected with the Earth into those rotating with the Earth. The computation involves standard procedures available from geodetic and astronomical observatories in the frame of their software for orbit determination and prediction. We will use Czech software (Lála, 1981; for more details see Straka et al., 1991).

Satellite images usually include several kinds of distortions that must be corrected before the image data can be used for remote sensing applications. These errors can be divided into "orbital" and "geometrical" ones as well as those due to satellite hardware. A list of the errors that affect the navigation product is contained in Table 1 together with their classification. We display the magnitude of the errors before and after our improvement of navigation software. Some errors can not be removed by the users without a cooperation with the responsible space agency (Nos. 1, 6, 9); the error No. 11 is dictated dominantly by the pixel size. For more details see the review paper Straka et al. (1991) and the references in this paper. The mathematical background for the direct and inverse referencing has already been described in literature (Ho and Asem, 1989; Emery et al., 1989; Robinson, 1985), so we will repeat only the necessary minimum (Section 3) to understand the meaning of the changes made in the software (Sections 4.1, 4.2), and we will deal with the software alidation test (Section 5).

Table 1: Classification of errors in satellite navigation (for NOAA-N type orbit).

No	error	error classes		error type/ comment
		before	after	
1	orbital elements of NASA two line type	2	2 &	OR
2	neglected orbital eccentricity (length of arc about 5000 km)	2	5	OR
3	neglected difference between geocentric and geodetic (geographic) latitude	1	5	max. at ± 45 deg zero at equator NO, G
4	neglected difference between astronomical and geodetic latitude	3	3 +	NO, G
5	neglected gravitational perturbations except Earth's polar flattening	2 - 4	3 - 4 +	see Table 4 OR
6	old (superseeded) main constants of the gravity field (GM) and ellipsoid parameters a_{el}, i_{el}	2	2 - 4	OR
7	neglected non-gravitational perturbation	3 - 4	4 +	OR
8	max. scan skew effect	2	5	NO, G
9	timing error	1	1 &	NO, H
10	attitude parameters	2	2	NO, H
11	identification of 1 x 1 km pixel on map	2	2 &	NO, G

OR = orbital
NO = non-orbital
H = hardware
G = geometric
& can not be improved by the ordinary users
\+ software for a further improvement is available but not used yet

Error classes

very important 1 error may be much larger than the pixel size, say > 10 km
Action:
if present software does not include such an effect, change it to include it.

important 2 error amplitude reaches the pixel size, i.e. ~1 km
Action:
include it, provided that new software becomes not extremely complicated and CPU-time consuming.

moderate	3	with amplitude A within the range (roughly) 100 m ≤ A < 1 km Action: usually no action, with the exception that improvements can be achieved easily.
small	4	10 m ≤ A < 100 m usually no action.
negligible	5	A < 10 m, no action.

2. *Orbital elements of meteorological satellites for navigation purposes*

We consider the orbits of the operational meteorological satellites NOAA 6 through 11; their basic orbital data and COSPAR identification numbers are gathered into Table 2.

Table 2: COSPAR identification numbers of the NOAA-N satellites, their approximate initial orbital characteristics, and list of periods where the NASA two-line or TBUS elements were available for our analyses (June, 1991 status):

satellite name	COSPAR number	P	I	apogee [km]	perigee [km]	NASA two-line elements	TBUS
NOAA 6	1979 - 057 A	100.9	98.5	809	790	1/81 - 8/86	6/80 - 8/80, 6/82 - 8/83, 7/84 - 11/86
NOAA 7	1981 - 059 A	101.8	99.0	851	832	1/82 - 3/87	6/81 - 3/85
NOAA 8	1983 - 022 A	101.1	98.5	822	797	7/83 - 12/84, 8/85 - 2/87, 8/87	4/83 - 8/84, 5/85 - 11/85
NOAA 9	1984 - 123 A	101.9	99.1	861	839	1/85 - 12/87	12/84 - 11/88
NOAA 10	1986 - 073 A	101.1	98.6	821	804	-----	9/86 - 12/90
NOAA 11	1988 - 089 A	102.0	98.6	861	845	-----	9/88 - 12/90

P orbital period in minutes
I orbital inclination in degrees

Sun-synchronous orbits:
NOAA even numbers twilight orbits (evening and morning),
NOAA odd numbers full-sun orbits (early afternoon and past midnight).

At the Max-Planck-Institut für Meteorologie, Hamburg, as for most users, only NASA two-line elements and TBUS elements are available for the NOAA-N satellites.

NASA computes the NASA two-line elements for plenty of application satellites. A truncated Earth gravity field model (see NASA, 1974; Attachment C) is

used for the orbit computation, where only the zonal harmonic geopotential coefficients $C_{2,0}$ and $C_{3,0}$ are retained. The analytical theory for motion of a satellite in the Earth gravity field (Kozai, 1959) is cited in (NASA, 1974). Kozai has extended his formulae to long-periodic perturbations due to $C_{3,0}$ through $C_{14,0}$. The simplified force model used in this orbit generation is adequate for the observations available, which are of low accuracy (radar, radio). Other orbit perturbations besides those due to the zonal geopotential harmonics seem to be omitted (ibid). The mean motion n is given among the elements instead of the semimajor axis a; the relationship to inverse the quantities is in Attachment B of NASA (1974). Also the derivatives of n are published; they can be used to model in a first approximation the drag effect causing secular decrease of a and of orbital eccentricity e. The NASA two-line elements enable position computation with a ±1 to ±5 km error compared to the actual position (NASA, 1974; NOAA 1991, priv. commun.).

The orbital elements called TBUS are available from NOAA/NESS (1981). We know only that these are Brouwer mean elements. The Brouwer and Brouwer-Lyddane analytical theory is an alternative theory describing satellite's motion in the Earth gravity field (see Cappellari et al., 1975); again only the zonal harmonics are kept (to $C_{5,0}$), all other perturbing accelerations are not accounted for. For our navigation purposes, when the pixel size is about 1 x 1 km^2, both simplified theories (Kozai or Brouwer-Lyddane) yield the similar accuracy, and the position of a satellite computed by them should be nearly identical (Lundquist and Veis, 1966). According to the technical documentation available at Max-Planck-Institut für Meteorologie, Hamburg, however, not only mean elements are distributed within TBUS files, but also actual (osculating) rectangular coordinates and velocity components, ballistic coefficient, daily solar flux values, radiation pressure coefficient and rates of the elements due to $C_{2,0}$ are added. This might be an indication that the force model used to compute the TBUS elements is more advanced (not only the zonal harmonics of the geopotential).

A comparison between these two types of orbital information revealed the following: (i) the epochs of the elements differ (thus a direct comparison between the argument of perigee w, longitude of the ascending node Ω and the mean anomaly of epoch M_o is not possible), (ii) when transforming n of the NASA two-lines to a by means of the recommended formula, this a differs by a few kilometers from a of the TBUS elements for the closest epochs, the difference being a constant for various epochs. The impact on navigation with these two types of elements is that the same software using the same force model will

yield different results. We have similar experience when comparing the NASA two-line elements to the Soviet elements transmitted to us to Ondrejov Observatory via Astrosoviet for the Intercosmos satellites. Table 3 compares basic geodetic parameters of the Earth used in the NASA predictions to their recent values. Although the value of the geocentric gravitational constant GM differs significantly, this difference fails to explain the much larger discrepancy in the semimajor axis mentioned above.

Table 3: Basic geodetic parameters of the Earth for satellite navigation software.

quantity	STEM 68R	WGS 72	WGS 84	GEM T2	GRIM 4
GM [km^3 s^{-2}]	398601.2	398600.8	398600.5	398600.436	398600.440
$10^{11}\ \omega_{Earth}$ [rad.s^{-1}]	-----	7292115	7292115	7292115	-----
a_{el} [km]	6378.145	6378.135	6378.137	6378.137	6378.136
$1/i_{el}$	298.25	298.26	298.257	298.257	298.257
$10^6\ C_{2,0}$	- 484.131	- 484.1605	- 484.16685	- 484.16530	- 484.16434

STEM 68R — Spacetrack Earth Model (NASA two-line elements, NASA, 1974)
WGS 72, 84 — World Geodetic System (see, e.g., Decker, 1986)
GEM T2 — Goddard Earth Model (T . . . TOPEX), Marsh et al., 1989
GRIM 4 — Earth gravity field model worked out in Germany and France (Schwintzer et al., 1990)
GM — G times M, the product of the universal gravitational constant with mass of the Earth (including the atmosphere), also called geocentric gravitational constant and directly derivable from satellite dynamics data
ω_{Earth} — angular rotation of the Earth
a_{el}, i_{el} — semimajor axis and the polar flattening of the reference rotational ellipsoid best-fitting geoid shape
$C_{2,0}$ — the most important of the harmonic geopotential coefficients (second degree zonal harmonic coefficient).

Various types of orbital elements are disseminated by various space agencies. Although the actual satellite observations are provided mostly by American and Soviet military services, the orbit determination, prediction and ephemeris data are often under the management of operational space agencies (e.g. NESS, NESDIS, ASTROSOVIET, etc.). Various examples of orbital element transmissions are listed in Smith (1980).

Smith (1980) gave an illustrative example. NORAD (North American Air Defense Command) of the US-Airforce performs satellite tracking and orbit determination. The NORAD force model for the orbit computation, although classified, was known to contain the Earth gravity and lunisolar perturbations, atmospheric drag and solar radiation pressure. The Earth gravity model used

was worked out by the US Defense Mapping Agency in the frame of WGS-72 (World Geodetic System) and may be it was replaced by the more recent WGS-84. From orbital elements, computed by NORAD, ephemeris data are generated and transmitted to NESS (National Environmental Satellite Service), where in turn, orbital elements are retrieved, computed by means of the NASA orbit determination programme (Cappellari et al., 1976). Thus, NESS provided the users with orbital elements that modelled the ephemeris NORAD data, not original tracking data! One can only hope that this complicated procedure, dimming the matter, has already been or will be superseeded by a direct approach.

The same force model that was used for the orbit computation should be used for the satellite navigation. We see, however, it may happen that the original agency computing the elements and/or the perturbation model used is not known. In such a case, some error in the navigation product will appear. To overcome this problem, community of users should press on space agencies to distribute well documented, clearly defined orbital elements, with a reliable accuracy assessment to know how large an error in the navigation result is due to the "orbital dynamics" imperfections, amended by a list of formulae necessary for the navigators to write their navigation software properly.

Another information which would be useful but is missing for NASA two-line and TBUS element sets concerns the orbit manoeuvres. Before using the elements we plot them as a function of time. The individual outlying points like those in Fig. 2a are probably due to a gross error in the data or a misprint in data transmission, while an offset, like that in the mean motion on Fig. 2b, is probably due to an orbit maneouvre.

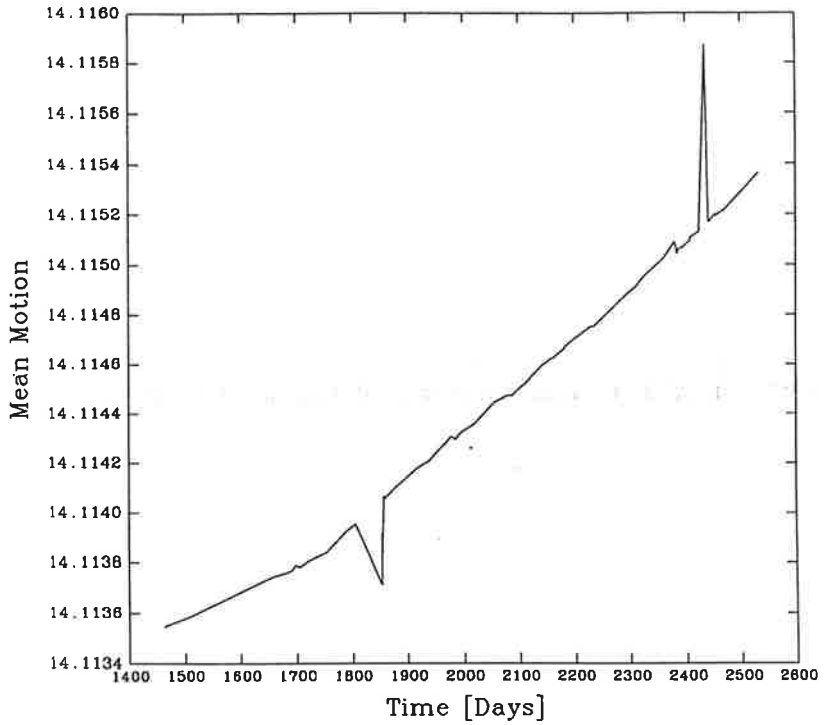


Fig. 2a: NASA two-line elements mean motion for NOAA 9 as a function of time, starting Jan. 1, 1981. Please note the errors in transmission.

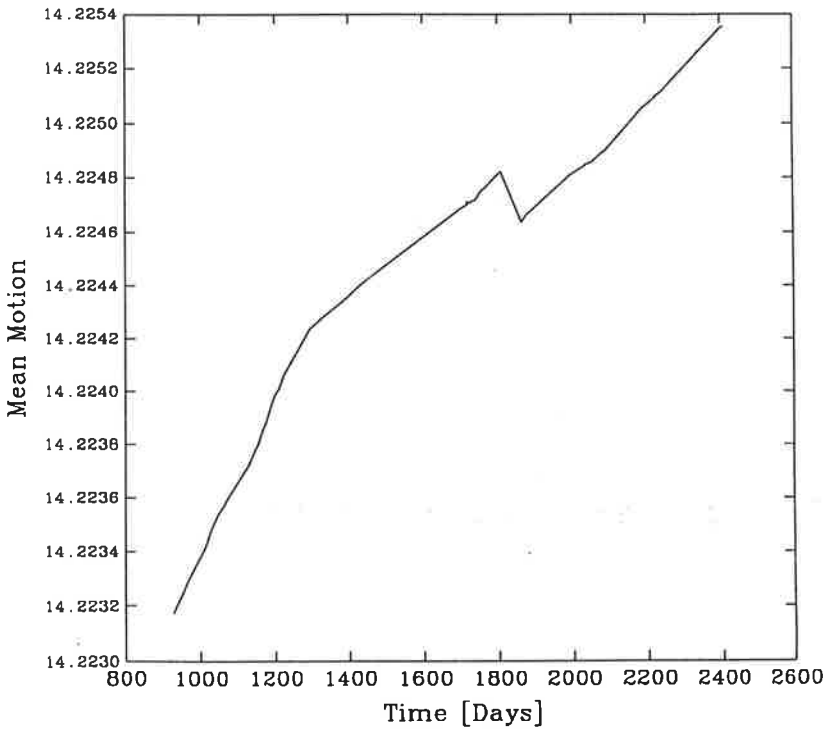


Fig. 2b: As Fig. 2a, but for NOAA 8; please note the change around day 1800 due to a manoeuvre.

3. *Procedures of direct and inverse referencing without ground control points*

In both cases, we assume the following input:

- orbital elements valid for the epoch t_0 not too far from time of the scene (max. 2 weeks),
- the scan step angle δ_i of the scanning radiometer
- the half scan line time $t_{1/2}$ and
- the line rate $\tilde{u}$ (number of lines scanned per second).

Procedures described in Ho and Asem (1986), and the programme PIXPO, available to us for inverse referencing, need also precise nodal information (time of equator crossing t_{eq} at geographic longitude λ_{eq}). Our procedures "EPHEM" and new "PIXPO" do not need this information.

In the case of direct referencing, pixel number p and scan line number l are given and the parameters sought are: The geodetic (geographic) latitude and longitude (φ, λ) of the centre of the pixel (at time t) at sea level with height $h = 0$ above a reference ellipsoid. The solution is straigthforward.

In the case of inverse referencing, φ and λ of the desired pixel, as taken from a map, are given, and the pixel indices l and p are computed. The solution is iterative.

As we already know from Section 1, mean orbital elements in t_0 will be transformed to osculating elements in t, and then they will be "rotated" to the coordinate system (φ, λ, h) connected with the Earth's body. A simplified force model will be used. For more details on its choice see Straka et al., 1991.

The most important effect of the Earth gravity field is the precessional motion, i.e. secular changes of ω, Ω, M_0 due to $C_{2,0}$, because $C_{2,0}$ is 1000 times larger than all the other harmonics C_{lm}, S_{lm}. According to textbooks, the variations of ω, Ω and M_0 are directly proportional to $C_{2,0}$ (e.g. Kaula, 1966; Vanícek and Krakiwski, 1986):

$$\dot{\omega} = \frac{3}{4} n C_{2,0} \sqrt{5} \left(\frac{R}{a}\right)^2 [1 - e^2]^{-2} [1 - 5 \cos^2 I] \quad , \tag{1}$$

$$\dot{\Omega} = \frac{3}{2} n C_{2,0} \sqrt{5} \left(\frac{R}{a}\right)^2 (1 - e^2)^{-2} \cos I \quad , \tag{2}$$

$$\dot{M}_o = -\frac{3}{4} n C_{2,0} \sqrt{5} \left(\frac{R}{a}\right)^2 (1 - e^2)^{-3/2} [3 \cos^2 I - 1] \quad , \tag{3}$$

where $10^6\, C_{2,0} = -484.1653$ or $-J_2 = C_{2,0}\sqrt{5} = 1082.6265$ if using the older notation; $R = 6378.137$ km (we take $R = a_{el}$), I = orbital inclination, and e = orbital eccentricity.

The formulae for the long-periodic perturbations of I, e, ω, Ω, M_o due to $C_{l,o}$ (odd degree zonal harmonics) are more complicated (see, e.g., Lundquist and Veis, 1966); as are those for the short-periodic variations caused by $C_{2,0}$ (ibid, Kozai, 1959; Cappellari et al., 1976).

The effect of atmospheric drag on a and e can be computed from the first derivative of n, $\dot{n}$ (NASA, 1974), as follows:

$$\dot{a} = -\frac{4}{3}\left(\frac{a}{n}\right)\left(\frac{\dot{n}}{2}\right) \quad , \tag{4}$$

$$\dot{e} = -\frac{4}{3}\left(\frac{1 - e}{n}\right)\left(\frac{\dot{n}}{2}\right) \quad . \tag{5}$$

The osculating element E at time t is computed from its mean value at t_0 by:

$$E_t = E_o + \dot{E}_o (t - t_o) + \sum_i \delta E_{oi} \quad , \tag{6}$$

where $\dot{E}_o$ are the secular perturbations and the δE_{oi} are the individual periodic perturbations. For the mean anomaly, the mean motion n has to be added, leading to

$$M = M_o + \dot{M}_o (t - t_o) + \sum_i \delta M_{oi} + n (t - t_o) \quad , \tag{7}$$

where n can be computed from the semimajor axis a of the orbital ellipse by means of the 3rd Keplerian law

$$n^2 a^3 = GM \quad , \tag{8}$$

where GM is the geocentric gravitational constant (see Table 3), presently known from of satellite dynamics to at least 7 digits.

The orbital elements at time t can be transformed to rectangular geocentric coordinates related to the vernal equinox (not rotating with the Earth), then rotated by the sidereal Greenwich time Θ to the coordinates rotating with the Earth (see Lundquist and Veis, 1966; and many others; also the programme ORBITS, Section 4.1.1). A direct transformation is also possible (Colombo, 1984)

$$\Phi_S = \arcsin\left[\sin I \sin(\omega + v)\right] \quad , \tag{9}$$

$$\lambda_S = \arcsin\left[\cos I \sin(\omega + v) / \cos \Phi_S\right] + \Omega - \Theta \quad , \tag{10}$$

where Φ_S, λ_S are geocentric (not geodetic!) latitude and longitude (for more details see Section 4.1.2).

The sidereal angle Θ rotates the "celestial" coordinate system, which is defined in fact by the orbital elements to the true terrestrial coordinate system which is related to the Earth's equator, rotational pole and Greenwich meridian. Θ is a function of precession and nutation. Standard procedures are available to compute mean or true Θ; we make use of that from PRIOR (Lála, 1981) or EPHEM 5 (Kostelecky, 1990). In the programme ORBITS, this "rotation" is performed by the function VERNEQ.

The true anomaly v in (9) and (10) can be found from the mean anomaly M by means of the equation of centre:

$$v = M + 2\,e \sin M + \frac{5}{2} e^2 \sin 2M + 0\left(e^3\right) \quad . \tag{11}$$

For NOAA-N satellites $e \sim 0.001$, so only the first two terms in (11) are important.

The mean anomaly M can be computed by (7), as M_0, a or n are given, and the necessary perturbations are already accounted for. The oversimplification, namely a circular orbit from the time of equator crossing t_{eq} to time t as used by Ho and Asem (1986) is thus overcome; "our" orbit is a perturbed elliptical one.

The next step is to convert the geocentric coordinates Φ_s, λ_s of the subsatellite point S (Fig. 3), computed by (9) - (11), to geocentric coordinates (Φ_s, λ_s) of the pixel D. First we find the angle Ψ from the off-nadir angle δ

$$\Psi = \arcsin\left[\left(r_p / r_{el}\right) \sin\delta\right] \quad , \tag{12}$$

for the sign see Emery et al. (1989).

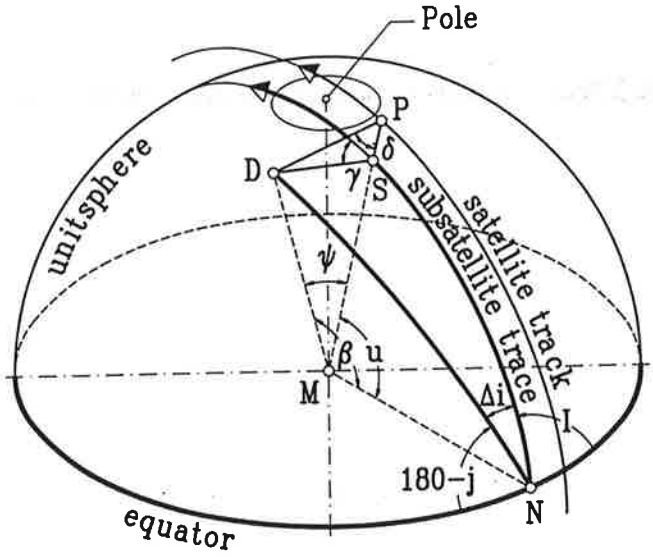


Fig. 3: Schematic for direct and inverse referencing: P is the position of the satellite, S the subsatellite point and D the desired pixel position. M is mass center of the Earth and N the ascending node of satellite orbit. The angle Δi is the difference between orbital inclination of the pixel and of satellite orbital inclination I.

The radius vector of the satellite orbit $r_p \equiv PM$ at time t and the geocentric distance DM of the pixel (approximated by $SM \approx DM \approx r_{el}$ i.e. by the radius-vector of a reference ellipsoid computed for S, because Φ_S is already known) follow (Fig. 4) from

$$r_p = a\left(1 - e\cos\overline{E}\right) = \frac{a\left(1 - e^2\right)}{1 + e\cos\nu} \cong a\left(1 - e\cos M\right) \quad , \tag{13}$$

$$r_{el_D} \cong r_{el_S} = \sqrt{\frac{a_{el}^2\left(1 - e_{el}^2\right)}{1 - e_{el}^2\cos^2\Phi_S}} \cong a_{el}\left(1 - 2\,i_{el}\sin^2\Phi_S\right) \quad , \tag{14}$$

with

$$e_{el}^2 = \frac{a_{el}^2 - b_{el}^2}{a_{el}^2} = 2\,i_{el} - i_{el}^2 \quad ,$$

where E is the eccentric anomaly, r_{el}, a_{el}, e_{el}, i_{el} are the radius-vector, semimajor axis, eccentricity or flattening of the reference rotational ellipsoid, respectively. Recent values of these parameters should be used (Table 3).

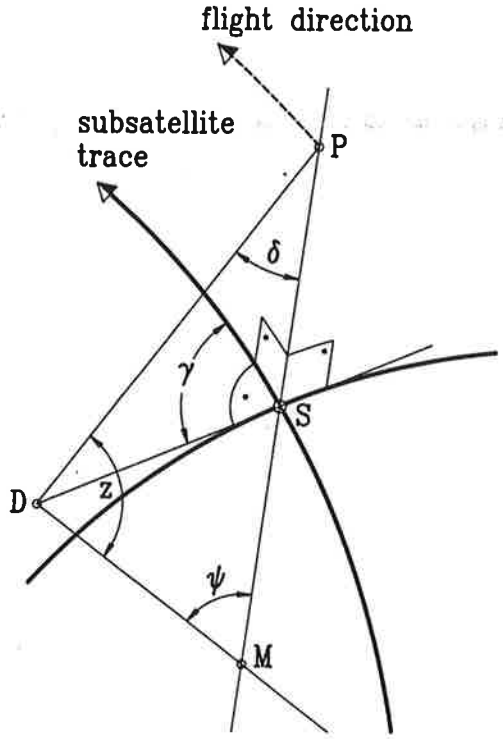


Fig. 4: Enhanced scheme for direct and inverse referencing. The relationship between the off-nadir viewing angle δ from satellite P and the geocentric angle ψ of the pixel D.

Now, we have the angle $\psi = DS$ in the triangle DMS (Fig. 3). Also the angle u = SN is known; it is the so-called argument of latitude $u = w + v$; $\underline{w}$ is given, v is computed by eqs. (7) and (11). Thus we can find $\beta = DN$ in the triangle DSN (Fig. 3) from

$$\cos \beta = \cos \psi \cos u - \sin \psi \sin u \cos \gamma \ , \qquad (15)$$

where the angle γ is close to 90°, but slightly lower than 90°, because of the scan skew effect (Fig. 5) which is due to the advancement of the satellite during the time needed for a scan:

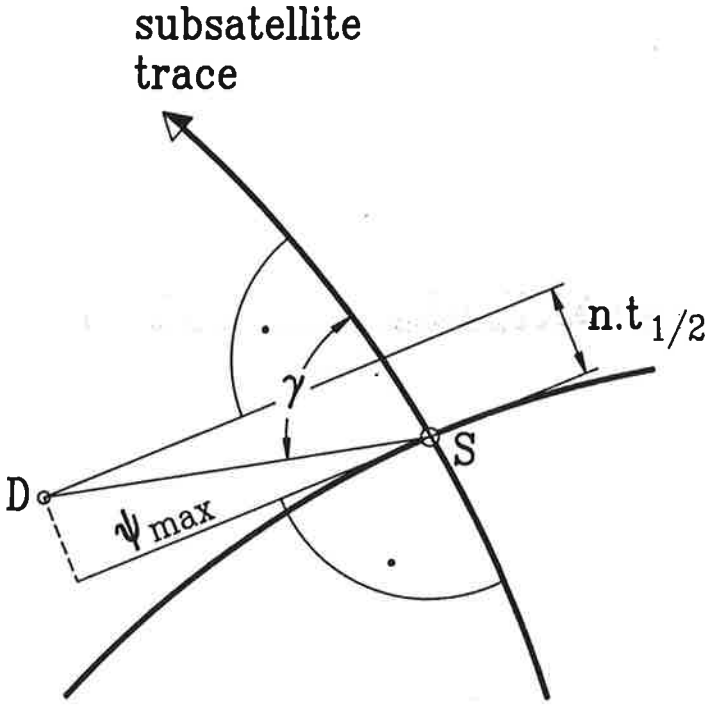


Fig. 5: The influence of the scan skew effect on direct/inverse referencing.

$$\cos \gamma = \frac{n\, t_{1/2}}{\psi_{max}} \quad , \tag{16}$$

ψ_{max} is the maximum of ψ, and the mean motion n is orbit dependent, the time needed for a scan, $t_{1/2}$, is a satellite-hardware dependent quantity.

From the triangle DSN in Fig. 3, we get the inclination difference between the satellite orbit and the pixel position D

$$\sin (j - I) = \sin (\Delta i) = \frac{\sin \psi \sin \gamma}{\sin \beta} \quad . \tag{17}$$

From the triangle DL_DN (Fig. 6), the geocentric latitude Φ_D of the pixel D is simply

$$\sin \Phi_D = \sin \beta \sin j \quad , \tag{18}$$

and

$$\cos\beta = \cos\Phi_D \cdot \cos\left(L_D N\right) + \sin\Phi_D \cdot \sin\left(L_D N\right) \cdot \cos 90° \ .$$

This allows the calculation of the geocentric longitude λ_D

$$\cos\left(L_D N\right) \equiv \cos\Delta\lambda = \cos\beta \,/\, \cos\Phi_D \ . \tag{19}$$

Since from Fig. 6 the geocentric longitude λ_D is directly related to $\Delta\lambda$

$$\Delta\lambda = \left(\Omega - \Theta\right) - \lambda_D \tag{20}$$

we have both pixel coordinates (Φ_D, λ_D). For the transformation from geocentric to geodetic coordinates see Section 4.1.3.

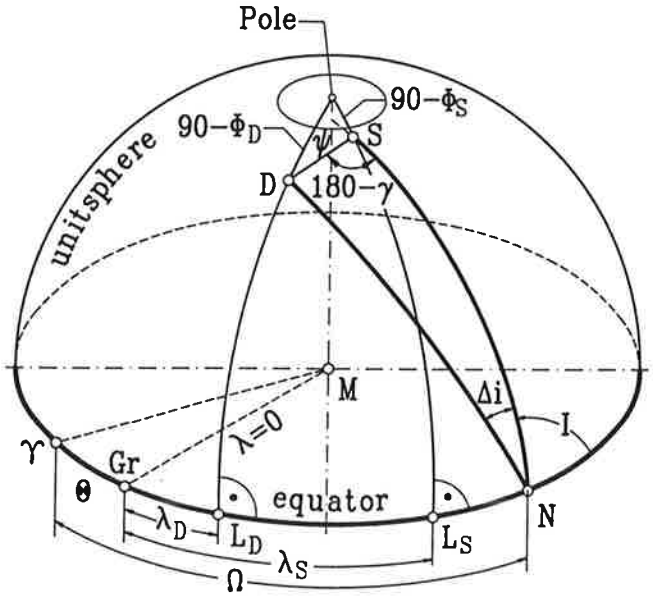


Fig. 6: A schematic for the angle relations for direct and inverse referencing. Here: transformation between orbital and terrestrial coordinate systems.
γ vernal equinox (first point of Aries)
Gr Greenwich meridian

For direct referencing we can apply the formulae just given, while for the inverse referencing, t is not known, and therefore, all time-dependent quantities must be computed iteratively; the first approximation may take a circular orbit as in Ho and Asem (1986).

4. *Software for direct and inverse referencing*

4.1. *Direct referencing*

4.1.1. *The programme ORBITS*

The programme ORBITS distributed by the University of Wisconsin, Madison, only computes the subsatellite (but not general pixel) position from mean elements at time t_0 for the required time interval Δt and space window, the latter given as geocentric latitudes and longitudes Φ_{min}, Φ_{max}; λ_{min}, λ_{max}. The latitude given by the programme ORBITS is the geocentric latitude Φ_S of the subsatellite point.

The central portion of ORBITS applies the Brouwer theory via the Brouwer-Lyddane orbit generator routine BROLYD. It has been written to transform mean to oscculating orbital elements. The Brouwer theory is an analytical theory for satellite motion around the Earth as disturbed by the first few zonal harmonic geopotential coefficents only (for details see Sections 5.9 - 5.10 of Cappellari et al. (1976), where also the original publications of Brouwer and Lyddane are cited).

The present version of ORBITS reads a window for time, latitude and longitude and seeks automatically the most recent orbital (TBUS) elements of the required satellite. The improved version of ORBITS, called ORBITEST, described in Section 4.1.4, does not differ in the input and the user may work with both programmes in the same way.

4.1.2. *Geocentric, geodetic and astronomical latitude*

Which subsatellite point belongs to the satellite position P_i at time t_i (Fig. 7a, b)? Is it P_S, the projection of P_i onto a spherical Earth, or is it P'_S, the projection of P_i onto a geocentric ellipsoid best-fitting the Earth' shape, or is it P''_S, the projection of P_i onto one of the equipotential surface, called the geoid? In nature, it is a point P'''_S, (not shown in Fig. 7a, b) somewhere on land or sea-surface topography, near P''_S. For us, the subsatellite point will be approximated by P'_S. It will be shown that the difference between P_S, and P'_S, may be very significant even for a pixel size of about 1 km, while the difference between the ellipsoidal position P'_S, and the "geoidal" position P''_S, is much lower (see Table 1). Finally, the difference between P''_S and P'''_S can be neglected.

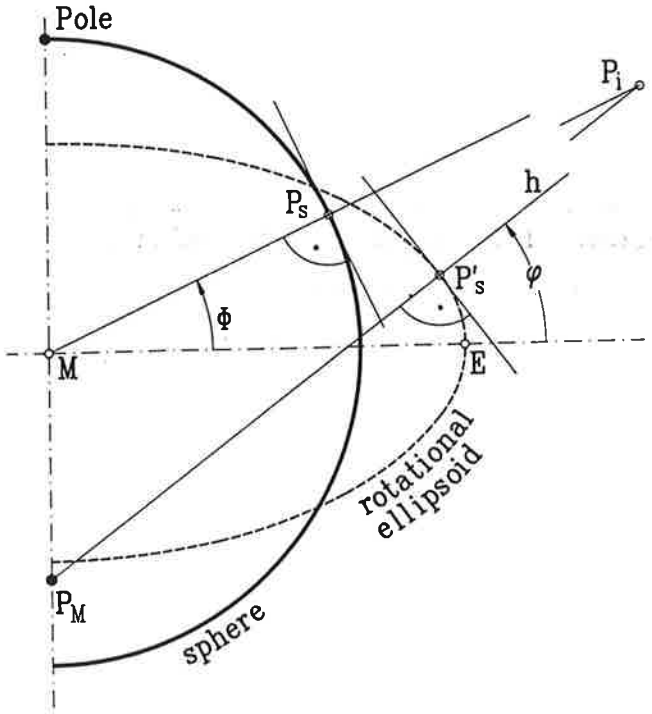


Fig. 7a: Geocentric Φ and geodetic φ latitude. The maximum difference is about 11 arcmin (~20 km for points on Earth surface). M is the mass center of the Earth as well as the center of a reference (= thus geocentric) ellipsoid.

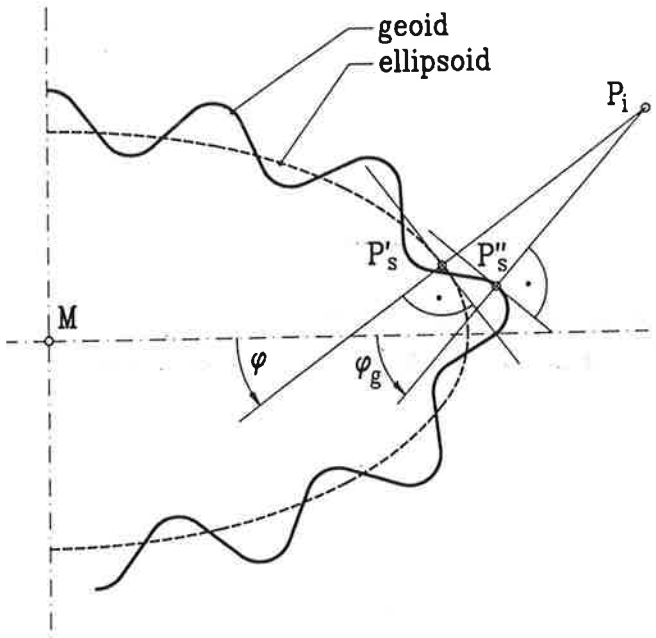


Fig. 7b: Geodetic φ and astronomical φ_g latitude. The maximum difference is a few arcseconds (~100 meters on Earth surface). The undulations of the geoid over the reference ellipsoid are plotted enhanced (geoid is always convex).

The latitude Φ, related to P_S is called geocentric latitude, the latitude φ, related to P'_S is the geodetic (geographic) latitude, and the latitude φ_g, related to P''_S, is the astronomical latitude. We will deal with the geodetic latitude φ, related to a best-fitting geocentric rotational ellipsoid, characterized by its semimajor axis a_{el} and eccentricity e_{el} or flattening i_{el} (Tables 3, 4).

As for the longitude, there is no difference between the geocentric and geodetic longitude λ, while the astronomical longitude λ_g differes from λ by a deflection of vertical. The height h, above the reference ellipsoid, also differs from the height above the geoid due to the geoid undulation N (Fig. 8). The differences between (φ, λ, h) and $(\varphi_g, \lambda_g, h_g)$ are of the order of 10 to 100 meters, so their omission would lead to an error of category "3" (Table 1), still acceptable for present-day meteorological satellites with 1 x 1 km^2 pixel size. For more information on this topic see any textbook on geodesy, e.g. Vanícek and Krakiwski (1986). In the Astronomical Institute of the Czech Academy of Sciences, a software to account for the difference for any place on Earth is available (Síma et al., 1983). For the "Hamburg area", N ≈ 40-50 m, and fortunately the geoid is flat so N ≈ const for all German coastlines.

Maps are usually related to an ellipsoid, thus the geodetic latitude is also called geographic latitude. The geodetic coordinates of a pixel (φ_D, λ_D), scaled from a map, are sometimes used to define Ground Control Points (GCP). We will not use them for any orbit "rectification"; we will use them for comparisons and software validation only. The geographic latitudes for this purpose should be derived from recent high-quality maps to avoid the use of old reference ellipsoids (compare Table 4 to 3). We make use of INTERNATIONALE KARTENSERIE NORDSEE in European Datum related to WGS 84 (Table 3). The accuracy of scaling from those maps is limited, in fact, by the pixel size in a satellite image (1 km). Assuming that a "standard set" of GCPs in the area of interest is used repeatedly, one can recommend to contact a geodetic survey (service) for the very precise geodetic or astronomical coordinates ("absolute landmarks"); they are known from triangulation, trilateration and by GPS measurements for any place on the Earth with at least a meter accuracy, usually with a centimeter accuracy, which is by a few orders higher than is necessary for our purpose. By this way, the user can achieve the best accuracy available even for a very small pixel size, where the reading from ordinary maps would not be accurate enough.

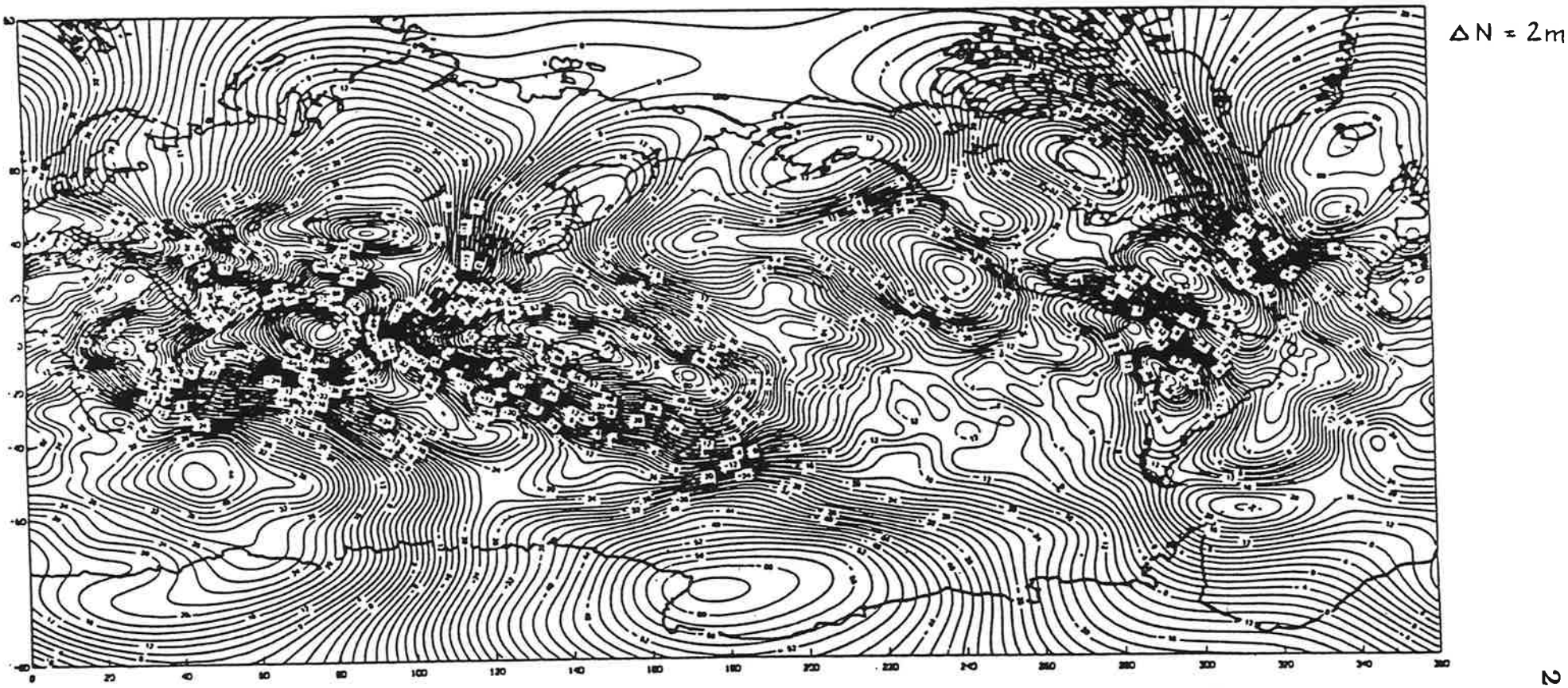


Fig. 8: Geoid undulations N [m] with respect to the best-fitting geocentric rotational reference ellipsoid with a_{el} = 6378.137 km, i_{el} = 1/298.257. Reproduced from Marsh et al. (1987).

It is evident that the geocentric coordinates (Φ_D, λ_D) of a pixel, computed by (1) - (20), must be transformed to geodetic coordinates (φ_D, λ_D) before comparing them with the "map" coordinates of the GCPs. The formulae follow.

Table 4: A historical review of values for the main parameters of a rotational reference ellipsoid for the Earth (see also Table 3).

Reference	a_{el} [km]	$1/i_{el}$
Delambre-Mèchain (1810)	6376.985	308.64
Bessel (1841)	6377.397	299.15
Hayford (1910)	6378.388	297.0
Krasovskij (1940)	6378.245	298.3
IUGG (1975)	6378.140	298.257
WGS (1984)	6378.137	298.257

4.1.3. Transformations between geodetic and geocentric latitudes

The rectangular (Cartesian) coordinates of the position of a satellite or subsatellite point P_i (or P_s) and its spherical coordinates (both geocentric *) are related as follows:

$$\begin{aligned} X &= r \cos\Phi \cos\lambda \\ Y &= r \cos\Phi \sin\lambda \\ Z &= r \sin\Phi \end{aligned} \quad , \tag{21}$$

with $r = MP_s$.

The rectangular coordinates of P_i (or P'_s) and its geodetic coordinates (both geocentric *) can be transformed by this way:

$$\begin{aligned} X &= (r_N + h) \cos\varphi \cos\lambda \\ Y &= (r_N + h) \cos\varphi \sin\lambda \\ Z &= \left[r_N (1 - e_{el}^2) + h\right] \cdot \sin\varphi \end{aligned} \quad , \tag{22}$$

where r_N is the normal radius of curvature (also called prime vertical radius of curvature) of the ellipsoid,

---

* The term "geocentric" here means "related to the Earth mass center" (M in Fig. 7a, b). So we should be strict and distinguish "geocentric geocentric" from "geodetic geocentric" coordinates, while "geodetic non-geocentric" coordinates would be related to a non-geocentric ellipsoid.

$$r_N \equiv r_N(\varphi) = a_{el} / \left(1 - e^2_{el} \sin^2 \varphi\right)^{1/2} \qquad (23)$$

and h is the height above the ellipsoid. The problem is that r_N is a function of φ and not a function of Φ.

By comparing (21) and (22), we get (24):

$$\operatorname{tg} \Phi = \left[\frac{r_N\left(1 - e^2_{el}\right) + h}{r_N + h}\right] \cdot \operatorname{tg} \varphi \quad , \qquad (24)$$

and

$$\operatorname{tg} \lambda = Y/X \quad .$$

For small h, equations (24) becomes

$$\operatorname{tg} \Phi = \left(1 - e^2_{el}\right) \cdot \operatorname{tg} \varphi = \frac{b^2_{el}}{a^2_{el}} \cdot \operatorname{tg} \varphi \quad , \qquad (25)$$

where b_{el} is the semiminor axis of the reference ellipsoid. Thus, equation (25) is identical to equation (3) in Emery et al. (1989) for small values of h, i.e. not for the satellite position P_i, where $h \approx 850$ km.

We see, the transformation from geodetic to geocentric latitude is direct. The inverse transformation is more complicated and is solved usually by iterations, although an analytical solution is also known (see Vanícek and Krakiwski, 1986, p. 326).

The difference $\varphi - \Phi$ is zero for $\varphi = 0°, \pm 90°$, but at the poles the formulae (24) and (25) fail; the maximum difference $(\varphi - \Phi)_{max} \approx 11$ arc min (≈ 19 km for points on the Earth surface) for $|\varphi| = 45°$.

4.1.4. ORBITEST-improved ORBITS

The improved programme ORBITS, called ORBITEST, differs just by the transformation from geocentric to geodetic latitude of the pixel at the end of computation. The subroutine GEOGEO is called and φ_D is computed from Φ_D by the following iterative process with prescribed accuracy ε:

1. first iteration:

$$p = (X^2 + Y^2)^{1/2} \quad , \quad \varphi_o = \arctan\left[\frac{Z}{p\left(1 - e_{el}^2\right)}\right] \quad ,$$

where (X, Y, Z) of P_i, eq (21), are known from previous computation in ORBITS (ORBITEST),

$$r_{N,o} = a_{el} / \left(1 - e_{el}^2 \sin^2 \varphi_o\right)^{1/2} \quad , \quad h_o = 0 \quad \text{or} \quad \left(p / \cos \varphi_o\right) - r_{N,o} \quad .$$

2. k th iteration:

$$\operatorname{tg} \varphi_k = \left[\frac{r_{N,k-1} + h_{k-1}}{r_{N,k-1}\left(1 - e_{el}^2\right) + h_{k-1}}\right] \operatorname{tg} \Phi \quad .$$

3. We stop if

$$\left|\varphi_{k+1} - \varphi_k\right| < \varepsilon \quad .$$

For our accuracy requirement, $\varepsilon = 10^{-4}$ [deg] and usually 2 iterations are sufficient. The user may change ε in the subroutine GEOGEO.

The user of the ORBITEST programme can see only one difference between ORBITEST and ORBITS output: ORBITEST yields time t, Φ, φ and λ of the subsatellite points (see example in Table 5) while the old ORBITS programme yielded only time, Φ and λ. The input to both programmes is identical. The subroutine *GEOGEO* is detailed in appendix 1.

Table 5: Geocentric and geodetic latitude of subsatellite points in degrees for NOAA-11 between 4 h 24 m and 4 h 42 m UT on November 20, 1990 as computed by the improved version of the programme ORBITS (taken from Straka et al., 1991).

h m time	Φ geocentric latitude	φ geodetic latitude	λ longitude
424	79.9042	79.9630	30.7883
425	77.9016	77.9714	15.3183
426	75.3176	75.4011	4.6058
427	72.4065	72.5047	- 2.8786
428	69.3055	69.4182	- 8.3115
429	66.0884	66.2148	- 12.4246
430	62.7962	62.9350	- 15.6603
431	59.4532	59.6026	-18.2913
432	56.0743	56.2325	- 20.4915
433	52.6693	52.8341	- 22.3761
434	49.2450	49.4141	- 24.0236
435	45.8055	45.9765	- 25.4894
436	42.3544	42.5248	- 26.8135
437	38.8941	39.0615	- 28.0256
438	35.4265	35.5883	- 29.1481
439	31.9529	32.1068	- 30.1985
440	28.4747	28.6184	- 31.1905
441	24.9925	25.1239	- 32.1351
442	21.5073	21.6244	- 33.0414

4.2. Inverse referencing

4.2.1. The programme PIXPO

The programme PIXPO (for pixel position) computes the pixel indices (l, p) for given geocentric coordinates Φ_D, λ_D ($h_D \equiv 0$), given satellite orbital parameters (I, a, t_{eq}, λ_{eq}), and given satellite "hardware" parameters scan line time $t_{1/2}$ and line rate $\dot{u}$. All symbols have been explained at the beginning of Sec. 3. The basis for PIXPO are the Ho and Asem (1986) formulae.

From the viewpoint of satellite dynamics, PIXPO is too much a simplification: between equator crossing time t_{eq} and time of scene the satellite is moving on a circular orbit around a circular Earth. The force (perturbation) model is strongly reduced, it accounts only for the nodal precession due to the most important zonal harmonic coefficient of the Earth's potential, $C_{2,0}$. The rate of $\dot{\omega}$ due to $C_{2,0}$ is neglected, because the orbit section from equator to scene is circular. The transformation from the inertial to the terrestrial system is achieved by rotation of λ due to the Earth rotation (e.g. (6) in Ho and Asem,

1986). Rather old constants GM and a_{el} are used. So we have verified, that just an update of the constants resulted in $\Delta l = 2$, i.e. an error of category 2, Table 1, was introduced.

The need to improve PIXPO for better navigation is evident. For example, the assumption of a circular orbit may result in a few kilometers error in the satellite and subsatellite position (error type 2, category 2, Table 1), depending upon the arc's length and geometry. From (13), we have for the radius of the perigee $r_p = a\,(1 - e)$ and the apogee $r_a = a\,(1 + e)$. The maximum position error due to the neglection of orbital eccentricity is $| r_p - r_a | = 2\,ae$, i.e. roughly 20 km for NOAA-N orbits. The other imperfections are due to the neglected polar flattening of the Earth (Section 4.1.2, and formulae (1) - (3) in Section 2), and due to other perturbing accelerations omitted, excluding $\dot{\Omega}\,(C_{2,\,0})$.

4.2.2. The programme EPHEM

The programme EPHEM (Lála, 1968) computes the ephemerides of artificial Earth satellites (Section 1, Fig. 1), i.e. it computes from given orbital elements (mean elements for epoch t_o) the satellite position via osculating elements in $t > t_o$. Not only the satellite observability for a given ground station $(\varphi, \lambda, h)_i$ is estimated but also the mutual simultaneous visibility from more than one station. EPHEM gives detailed ephemeris data like time and corresponding azimuth and elevation for direct use in satellite tracking.

The original version of EPHEM has been updated many times. The force model of a recent version includes gravity field perturbations due to the zonal and tesseral harmonics, according to theories of Kozai and Kaula and a simplified model for the atmospheric drag (NASA, 1974). The programme is able to read and work with various types of orbital elements including the NASA two-line elements. The TBUS elements, however, for the NOAA-N satellites, have never been used. Many subroutines of the PRIOR programme for orbit determination (Lála, 1981), worked out by a group in the Astronomical Institute of Czech Acad. Sciences, are used in the EPHEM, too. The latest modification for a personal computer has been prepared by Kostelecky (1990); it is named EPHEM 5 and contains a simplified force model. This programme has been implemented at the Hamburg institutes and has been modified for inverse referencing. The force model has been further simplified to be compatible with the NASA two-line elements (see NASA, 1974, by assuming that this description of the NASA mean elements is still valid). So we account firstly for the secular perturbations due to $C_{2,\,0}$, see equations (1) - (3), secondly for the long-periodic

ones caused by $C_{3,0}$ and thirdly for the atmospheric drag, according to equations (4) and (5) for the NASA two-line elements.

[If the TBUS elements supplied by NOAA are used, $a = 0 = e$, because $\dot{n}$ is not available. The force model used to compute the TBUS elements is also not known, see Section 2, so the navigation results with these two categories of elements differ].

The main argument of applying EPHEM for the inverse referencing instead of PIXPO is the following: Replace the term "satellite observing station" by "pixel" D (Fig. 3). Then we search for the moment, when D and S are on the same orbital meridian, $\gamma = 90$ (neglecting for a moment the scan skew effect), δ = min. This is the instant when the relevant line is scanned. It is the time of culmination of satellite P over pixel D. And the programme EPHEM just computes by an iterative process the time of such a culmination.

4.2.3. User guide for the programme PIXPO3 - a modified EPHEM 5

All subroutines tailored to the detailed ephemerides and a catalogue of satellite tracking stations were rejected from the EPHEM 5 version for a PC (Kostelecky, 1990), and various changes in the main of the programme and in the subroutine INEL (input of orbital elements) have been performed. The most important change was the inclusion of the subroutine LONGJ 3 which computes the longperiodic perturbations due to the zonal harmonic coefficient $C_{3,0}$ of the Earth potential, before the time of culmination is calculated. Also the format of input data to the original EPHEM 5 differs from that for the new version of EPHEM 5, called PIXPO 3. In Hamburg both programmes are available.

The user guide for PIXPO 3 will follow now with the NASA two-line elements with an example for an input in Table 6a, and of the results in Table 6b. A discussion of the results is in Section 5.

Explanation of input data in Table 6a:

(i) 1st line (free format):

- type of orbital elements — 2 for NASA two-line elements; 9 for TBUS elements
- option for computation of the longperiodic perturbations: 0 not included; 1 included

- half scan line time in [sec] 0.083 for subsatellite track

(ii) 2nd and 3rd lines

a) type 2 (formatted NASA two-line elements)

-- first line of NASA two-line elements:

column	format	name of varibale	meaning
1. - 9.	9 X	---	blanks
10. - 17.	A 8	SATE	name of satellite
18.	1 X	---	blank
19. - 20.	I 2	NY	last two digits of the year at the epoch of the elements
21. - 32.	F 12.8	DAY	day of the year, its fraction included
33.	1 X	---	blank
34. - 43.	F 10.8	EM(3)	$\dot{n}$ [rev/day^2]
44.	1 X	---	blank
45. - 50.	F 6.5	EM(4)	$\ddot{n}$ [rev/day^3] (usually zero)
51. - 52.	I 2	NX	exponent of $\ddot{n}$,

-- second line of NASA two-line elements:

column	format	name of varibale	meaning
1. - 8.	8 X	---	blanks
9. - 16.	F 8.4	EI(1)	inclination I [deg]
17. - 25.	F 9.4	EG(1)	Ω [deg]
26. - 33.	F 8.7	EE(1)	e (without first zero, see the example)
34. - 42.	F 9.4	EP(1)	ω [deg]
43. - 51	F 9.4	EM(1)	M_o [deg]
52. - 63.	F 12.8	EM(2)	n [rev/day]

These elements are available on a magnetic tape as files NOAAn.D, where n = 6-9, or TIROSN.D.

b) type 9 (unformatted TBUS elements), ordered exactly as in the input for ORBITS or ORBITESTS, occupying the 2nd and 3rd input lines.

(iii) 4th line (free format):

- required time for the inverse referencing (start of computation with PIXPO 3):

Y, M, D, H, M, SED (blanks between H & M & SEC);

- time correction between "rotational" UT1 and "coordinated" UTC time scales, taken for the required day from BIH Annual Reports - see appendix 2; (UT1-UTC) is in [sec]; for our purpose it may be neglected.

- rate of (UT1-UTC) in [sec/day]; here zero;
- integer number MJD (modified Julian date) for which the correction (UT1-UTC) applies (see also the BIH Annual Report in appendix 2).

(iv) 5th and other lines (free format):
- number of GCP or of pixel (for identification),
- geodetic latitude φ of the point [deg],
- longitude east of Greenwich (0-360°) [deg],
- height above the reference ellipsoid (here zero) in [m].

Our tests revealed that a scene with 262144 pixels is computed by means of PIXPO 3 during 1.7 h of CPU-time on SUN SPARC station 1 computer (0.3 h for the old PIXPO). The memory required for PIXPO 3 is about 295 KB (172 KB for the old PIXPO).

```
2   1   0.0833
        'NOAA09'  87233.52733660  0.00000162
        099.0565 199.7390.0016539 025.9284 334.2712  14.11513066
870819   14 43 10   -0.43    0.0   47029
3000        0.0    9.55   0.0
3001       55.0585 8.4335 0.0
3002       54.7417 8.2917 0.0
3003       55.5583 8.0833 0.0
3007       51.3917 1.4500 0.0
3008       58.1083 6.5667 0.0
```

Table 6a: Input data for PIXPO 3 or PIXPO 4.

Explanation of PIXPO 3 output in Table 6b:

First, the programme informs on satellite name and source of elements, then the elements used are reproduced (their rates are added) as follows:

ORBIT number of orbital arc (not important)
EPOCH (MJD and fraction of day, plus usual date in D, M, Y, H : M : SEC)
PERIG = ω [deg], $\dot{\omega}$ [deg/day],
RNODE = Ω [deg], $\dot{\Omega}$ [deg/day] ≈ 1 (because of nearly sun-synchronous orbits of NOAA-N satellites),
INCLIN = I [deg],
ECCEN =e, $\dot{e}$ [day^{-1}],
SAXIS = a [km], $\dot{a}$ [km/day],
MANOM = M_0 [rev], n [rev/day], $\dot{n}$ [rev/day^2), $\ddot{n}$ [rev/day^3].

```
SATELLITE : NOAA09'
SOURCE OF ELEMENTS :
                        NASA TWO-LINE ELEMENTS

ORBIT :            6
EPOCH :        47028    0.527337  21. 8.1987  12:39:21.88
PERIG :    25.928400   -2.810521
RNODE :   199.739000    1.009143
INCLN :    99.056500
ECCEN :     0.001654    0.000000
SAXIS :     7233.750      -0.001
MANOM :     0.928531   14.115131  0.800000D-06  0.000000D+00

LONGPER. TERMS (J3 ONLY) :
DP :  30.13486
DG :  -0.00001
DI :   0.00001
DE :   0.03032
DM :  -0.08371

SIDEREAL TIME AT 47028.5 : 10:36:34.428
CORRECTION (UT1 - UTC)   :     -0.43000     0.00000 47029
TSKEW (sec):  0.083

POINT No. FI(geod)     LAMBDA     H(m)    DATE        TIME CULM.   DELTA(deg)

 3000     0.000000   9.550000    0.0  19. 8.1987 14:43:21.21     0.02
 3001    55.058500   8.433500    0.0  19. 8.1987 14:58:43.88    46.79
 3002    54.741700   8.291700    0.0  19. 8.1987 14:58:38.53    46.50
 3003    55.558300   8.083300    0.0  19. 8.1987 14:58:52.62    46.38
 3007    51.391700   1.450000    0.0  19. 8.1987 14:57:49.55    28.54
 3008    58.108300   6.566700    0.0  19. 8.1987 14:59:37.10    44.74
```

Table 6b: Output of PIXPO 3 (data in Table 6a used).

```
SATELLITE : NOAA09'
 SOURCE OF ELEMENTS :
                        NASA TWO-LINE ELEMENTS

 ORBIT :            6
 EPOCH :        47028    0.527337  21. 8.1987  12:39:21.88
 PERIG :    25.928400   -2.810521
 RNODE :   199.739000    1.009143
 INCLN :    99.056500
 ECCEN :     0.001654    0.000000
 SAXIS :     7233.750      -0.001
 MANOM :     0.928531   14.115131  0.800000D-06  0.000000D+00

PROGRAM PIXPO4
STEP IN COMPUTATION [SEC]:     5.00
LONGPER. TERMS (J3 ONLY) :
DP :  30.13486
DG :  -0.00001
DI :   0.00001
DE :   0.03032
DM :  -0.08371

  3000     0.000000   9.550000   0.0  19. 8.1987 14:43:21.29     0.02
  3001    55.058500   8.433500   0.0  19. 8.1987 14:58:44.26    46.79
  3002    54.741700   8.291700   0.0  19. 8.1987 14:58:38.92    46.50
  3003    55.558300   8.083300   0.0  19. 8.1987 14:58:53.00    46.38
  3007    51.391700   1.450000   0.0  19. 8.1987 14:57:49.92    28.54
  3008    58.108300   6.566700   0.0  19. 8.1987 14:59:37.47    44.75
```

Table 6c: Output of PIXPO 4 (data in Table 6a used).

Second, the information on the longperiodic perturbations and on sidereal time, (UT1-UTC)-correction [sec] and $t_{1/2}$ = TSKEW [sec] follows.

Finally, the resulting time of culmination (D, M, Y, H : M : SEC) and the off-nadir angle δ [deg] are given for each point with (φ_D, λ_D, h_D). These values we then use to compute (l, p) of the desired pixels (φ_D, λ_D, h_D)$_i$ by means of well known formulae.

The pixel number p is computed from the off -nadir angle δ as follows:

$$p = (\delta_o - \delta)\frac{2047}{2\,\delta_o} + 1 \quad ,$$

where δ_0 = 55.38° for the AVHRR (max of δ). The pixel line l is related to the time of culmination t_c and to time of the the first line in the consideration t_f by this way:

$$l = \bar{u}(t_c - t_f) + 1 \quad ,$$

where $\bar{u}$ is the line rate. The values of p and l are computed outside PIXPO 3 (or PIXPO 4).

4.2.4. Non-standard use of PIXPO 3

Besides the standard use of PIXPO 3, i.e. to find time of culmination and the angle δ at the moment of culmination for the given pixel (φ_D, λ_D, h_D), PIXPO 3 can also be used to compute the time of equatorial crossing t_{eq}. Given: pixel (0, λ_D, 0), "fresh" orbital elements. We want to check t_{eq}, which was needed to run the old PIXPO (Section 4.2.1). We simply compute the time of culmination the satellite for the subsatellite pixel (0, λ_D, 0); this t^c_{eq} (c . . .computed) can be compared to t^o_{eq} (o . . .observed), for example from the University of Dundee satellite station data (the AVHRR observations). Sometimes, the value of t^o_{eq} is missing in the Dundee observations, so PIXPO 3 can amend it. Another possibility is to correct the observed value t^o_{eq} by means of t^c_{eq}, provided that the orbital elements used are of sufficient accuracy and the force model in PIXPO 3 is relevant to this accuracy.

4.2.5. PIXPO 4

The definition of the time of culmination used in EPHEM (the moment when D and S, see Fig. 3 or 5, are on the same orbital meridian) assumes that the local vertical and D and in S are in the same plane and intersect the mass center M. More generally, the time of culmination of P over D can be defined as a moment when the plane ρ and plane μ are perpendicular. The plane ρ is created by the unit vector of the local vertical in P and by the velocity vector of the satellite in P (both in the coordinate system not connected with the rotating Earth). The plane μ is given by the unit vector of the vertical in P and by the unit vector of the normal vector (product of the satellite velocity vector and satellite position vector, for point P). The mathematical solution makes use of formulae known from analytical geometry.

The geocentric position of D is known (given in input data), the geocentric position and velocity of satellite in P (and thus also in S) can be computed for arbitrary time t, in arbitrary small steps of $u = \omega + v$. The advantage of this method is that the satellite coordinates in t can be computed by means of any analytical theory (provided that its accuracy is sufficient for the given purpose), or even by a numerical integration of equations of motion with any force model. The orbital elements in t_0 are, however, the initial conditions in any case (no escape from the problems with the elements!).

The method just described is the basis of the PIXPO 4 programme. The main of PIXPO 3 has been replaced by a new main while the subroutines (coming mostly from the programme PRIOR), remain unchanged. PIXPO 4 is now in a test status; we used it for validation purposes (Section 5, Tables 8-9). Improperly chosen initial time or time steps Δu result in too CPU-time consuming computations. Probably the best is to choose the initial time close to the time of the first scan line of a scene. We have tested Δu = 0.1 to 10 sec and we found that the difference in the time of culmination and in the off-nadir angle due to the different time steps is negligible, so we recommend Δu = 5 - 10 seconds.

Information for users:

The input data to PIXPO 4 are identical to those to PIXPO 3, see Table 6a. The output of PIXPO 4 has a similar, but not the same form as the output of PIXPO 3, see Table 6c.

5. *Validation*

We consider the following tests of the new software:

(i) comparison to navigation with GCPs (Section 5.1),

(ii) intercomparison between (more or less) independent software (Section 5.2),

(iii) intercomparison of inverse referenced images of the same scene from different passages of the same satellite (Section 5.3).

5.1. *Validation by means of GCPs*

The purpose of an improved navigation is better geolocation of images retrieved from the Advanced Very High Resolution Radiometer (AVHRR) that is carried by the NOAA satellites of the TIROS-N series. Consequently, a validation of the inverse referencing is done by means of AVHRR observations.

The AVHRR measures radiation in five spectral channels distributed over the visible and infrared atmospheric windows. The ground resolution of the AVHRR measurements is 1.1 km at nadir view and 6.1 km at the largest scan angle of 55.38°. The scan direction is cross track at a sampling rate of 1 line (2048 pixels or resolution elements) per 0.16 seconds. The time of each scan line is measured by a satellite clock. It is given in milliseconds, however, the instrument manuals do not describe to which part of the scan line the time reference belongs to. While the scan is performed from the right to the left with respect to the flight direction of the satellite it covers a total scan angle of 110.76°. The corresponding pixel number (p) within a scan is used to compute the actual scan angle δ during the observation of pixel p in line l:

$$\delta = -\frac{p - 2049}{2048} \cdot 110.76° \quad , \quad 1 \le p \le 2048 \quad ,$$

where positive values indicate radiometer views to the right and negative values views to the left side.

A set of AVHRR measurements covering many pixels in adjacent scan lines can be put together forming a twodimensional image that allows an identification of coastlines if the scene is not obscured by clouds. Salient features of the coastlines might then be detected and located within single pixels. The geodetic positions of the salient features are known from sea-charts (based on the World Geodetic System WGS 72 and WGS 84 which also exhibit these structures).

We have chosen 15 salient features in coastlines of the North Sea acting as ground control points (GCPs). Their geodetic coordinates φ and λ are resolved from the maps with an accuracy of $\pm$ 0.25° leading to uncertainties of $\Delta Y \leq 0.46$ km in latitudinal direction and $\Delta X \leq 0.29$ km in longitudinal direction. The GCPs are listed in Table 7. Because of the AVHRR pixel size being 1.1 to 6.1 km in diameter we did not attempt to specify the location of the GCPs with higher accuracy.

Table 7: Ground Control Points used for validation of satellite navigation software for NOAA-N satellites.

name of point	geodetic latitude φ_N	longitude λ
Fair Isle SW-Edge	49° 31.0'N,	1° 39.0'W
Foula Southern Edge	60° 7.0'N,	2° 4.0'W
Esha Ness	60° 29.5'N,	1° 27.5'W
Sumburgh Head	59° 52,0'N,	1° 16.5'W
Utsira Western Edge	59° 18.5'N,	4° 52.0'E
Lista	58° 6.5'N,	6° 34.0'E
Thyboron	56° 42.0'N,	8° 13.0'E
Blaavands Huk	55° 33.5'N,	8° 5.0'E
Sylt Northern Edge	55° 3.5'N,	8° 26.0'E
Sylt Southern Edge	54° 44.5'N,	8° 16.5'E
Juist Western Edge	53° 40.0'N,	6° 52.5'E
Juist Eastern Edge	53° 41.0'N,	7° 6.0'E
Den Helder	52° 57.5'N.	4° 46.5'E
North-Foreland	51° 23.5'N,	1° 27.0'E
Out Skerries	60° 25.5'N,	0° 43.5'W

Eight passages of NOAA-7 and NOAA-9 have been selected and checked for clouds. The locations of the ground control points are then extracted in terms of line and pixel numbers (l, p) which are directly related to satellite clock time $t(l)$ and scan angle $\delta(p)$. A total of 44 points are examined. Their ($t(l)$, $\delta(p)$)-pairs are compared with the inverse referencing done by PIXPO, PIXPO3 and PIXPO4. The results are summarized in Table 8 and a simple statistics is reviewed in Table 9.

Table 8a/1: Satellite navigation, software validation.

Numerical results with NOAA-9				Day			AVHRR Observations Time of culm.			
point No.	φ (geod.) [deg]	φ (geoc.) [deg]	λ_E [deg]	D	M	Y	H	M	sec (UT)	δ [deg]
3000	0.0000	0.0000	9.5500	19	8	87	14	43	21.00	0.00
3001	55.0583	54.8776	8.4333	19	8	87	14	58	44.22	46.78
3002	54.7417	54.5601	8.2917	19	8	87	14	58	39.05	46.51
3003	55.5583	55.3786	8.0833	19	8	87	14	58	53.05	46.40
3007	51.3917	51.2039	1.4500	19	8	87	14	57	49.88	28.50
3008	58.1083	57.9354	6.5667	19	8	87	14	59	37.55	44.67
4000	0.0000	0.0000	12.9700	11	8	87	14	28	54.00	0.00
4001	55.5583	55.3786	8.0833	11	8	87	14	44	29.88	40.24
4002	59.3083	59.1391	4.8667	11	8	87	14	45	36.72	35.75
4003	59.8667	59.6993	358.7250	11	8	87	14	45	55.55	20.01
4004	60.4917	60.3264	358.3750	11	8	87	14	46	06.38	19.96
4005	60.1167	59.9502	357.9333	11	8	87	14	46	01.22	17.96
4006	60.4250	60.2595	359.2750	11	8	87	14	46	03.88	22.55
2000	0.0000	0.0000	10.2100	10	8	87	14	39	53.00	0.00
2001	55.0583	54.8776	8.4333	10	8	87	14	55	15.88	45.75
2002	54.7417	54.5601	8.2917	10	8	87	14	55	10.72	45.43
2003	55.5583	55.3786	8.0833	10	8	87	14	55	24.72	45.32
2004	53.6667	53.4828	6.8750	10	8	87	14	54	53.55	42.62
2005	53.6833	53.4994	7.1000	10	8	87	14	54	53.72	42.94
2006	52.9583	52.7731	4.7750	10	8	87	14	54	43.72	37.42
2007	51.3917	51.2039	1.4500	10	8	97	14	54	22.22	26.23

Table 8a/2:

Old PIXPO Programme Time of culm.			Updated EPHEM5 = PIXPO3 Time of culm.				New PIXPO (= PIXPO4) Time of culm.			
H	M	sec (UT)	H	M	sec	δ [deg]	H	M	sec (UT)	δ [deg]
---	---	---	14	43	21.29	0.02	14	43	21.29	0.02
14	58	46.36	14	58	43.00	46.79	14	58	44.26	46.79
14	58	40.99	14	58	38.60	46.50	14	58	38.92	46.50
14	58	55.14	14	58	52.62	46.38	14	58	53.00	46.38
14	57	51.82	14	57	49.60	28.54	14	57	49.92	28.54
14	59	37.79	14	59	37.10	44.74	14	59	37.47	44.75
---	---	---	14	28	54.11	0.04	14	28	54.20	0.03
14	44	31.22	14	44	28.73	40.33	14	44	29.10	40.33
14	45	37.94	14	45	35.17	35.80	14	45	35.51	35.81
14	45	56.97	14	45	54.13	20.07	14	45	54.64	20.07
14	46	07.95	14	46	05.97	20.04	14	46	05.39	20.04
14	46	02.66	14	45	59.79	18.01	14	46	01.11	18.01
14	46	05.16	14	46	02.28	22.63	14	46	02.61	22.63
---	---	---	14	39	51.40	0.02	14	39	51.49	0.02
14	55	18.75	14	55	14.61	45.80	14	55	15.00	45.81
14	55	13.40	14	55	09.29	45.48	14	55	09.68	45.49
14	55	27.54	14	55	23.37	45.39	14	55	23.75	45.39
14	54	56.12	14	54	52.08	42.55	14	54	52.46	42.56
14	54	56.20	14	54	52.16	43.00	14	54	52.54	43.00
14	54	46.32	14	54	42.32	37.56	14	54	42.70	37.57
14	54	24.98	14	54	21.07	26.32	14	54	21.44	26.33

Table 8b/1:

Numerical results with NOAA-9

point No.	φ (geod.) [deg]	φ (geoc.) [deg]	λ_E [deg]	Day D	M	Y	AVHRR Observations Time of Culm. H	M	sec (UT)	δ [deg]
5000	0.0000	0.0000	15.73000	12	8	87	14	17	57.00	0.00
5001	55.5583	55.3786	8.0833	12	8	87	14	33	35.88	33.80
5002	56.7000	56.5232	8.2167	12	8	87	14	33	54.88	35.10
5003	58.1083	57.9354	6.5667	12	8	87	14	34	20.88	32.18
5004	59.3083	59.1391	4.8667	12	8	87	14	34	43.05	28.99
5005	59.8667	59.6993	358.7250	12	8	87	14	35	04.05	10.92
5006	60.4917	60.3264	358.3750	12	8	87	14	35	14.88	11.03
5007	60.1167	59.9502	357.9330	12	8	87	14	35	09.88	8.76
5008	59.5167	59.3482	358.3500	12	8	87	14	34	59.22	8.87
8000	0.0000	0.0000	10.4900	09	1	87	14	20	23.00	0.00
8001	55.5583	55.3786	8.0833	09	1	87	14	35	58.68	44.46
8002	55.0583	54.8776	8.4333	09	1	87	14	35	49.68	44.83
8003	54.7417	54.5601	8.2917	09	1	87	14	35	44.35	44.56
8004	58.1083	57.9354	6.5667	09	1	87	14	36	43.52	42.78
6000	0.0000	0.0000	13.1800	10	1	87	14	09	28.00	0.00
6001	54.7417	54.5601	8.2917	10	1	87	14	24	52.35	39.32
6002	55.0583	54.8776	8.4333	10	1	87	14	24	57.35	39.75
6003	55.5583	55.3786	8.0833	10	1	87	14	25	06.35	39.26
6004	56.7000	56.5232	8.2167	10	1	87	14	25	25.68	40.13
6005	58.1083	57.9354	6.5667	10	1	87	14	25	51.35	37.59
6006	59.3083	59.1391	4.8667	10	1	87	14	26	13.68	34.72

Table 8b/2

Old PIXPO Programme Time of culm.			Updated EPHEM5 = PIXPO3 Time of culm.				New PIXPO (= PIXPO4) Time of culm.			
H	M	sec (UT)	H	M	sec	δ [deg]	H	M	sec (UT)	δ [deg]
---	---	---	14	17	56.81	0.05	14	17	56.90	0.05
14	33	37.52	14	33	34.67	33.91	14	33	35.03	33.92
14	33	56.17	14	33	53.24	35.15	14	33	53.60	35.16
14	34	22.12	14	34	19.09	32.27	14	34	19.44	32.27
14	34	44.78	14	34	41.65	29.08	14	34	41.99	29.09
14	35	05.78	14	35	02.58	10.98	14	35	02.91	10.99
14	35	16.75	14	35	13.51	11.14	14	35	13.83	11.15
14	35	11.67	14	35	08.45	8.81	14	35	08.77	8.82
14	35	00.96	14	35	57.78	8.92	14	34	58.10	8.92
---	---	---	14	20	23.38	0.70	14	20	23.47	0.71
14	35	56.73	14	35	59.65	44.50	14	35	59.00	44.49
14	35	48.98	14	35	49.86	44.92	14	35	50.19	44.92
14	35	43.65	14	35	44.53	44.59	14	35	44.85	44.59
14	36	42.50	14	36	43.34	42.80	14	36	43.72	42.81
---	---	---	14	09	28.83	0.28	14	09	28.92	0.28
14	24	50.98	14	24	52.21	39.35	14	24	52.58	39.35
14	24	56.22	14	24	57.44	39.82	14	24	57.82	39.82
14	25	05.07	14	25	06.29	39.36	14	25	06.68	39.36
14	25	24.35	14	25	25.56	40.20	14	25	25.98	40.20
14	25	49.97	14	25	51.16	37.63	14	25	51.60	37.64
14	26	12.25	14	26	13.43	34.79	14	26	13.89	34.80

Table 9: Software validation/comparison of numerical results.

Scene No.	Mean differences per scene [sec]			Age of orbital elements from epoch equator crossing *
	(1)	(2)	(3)	[days]
3	- 1.67	0.26	- 0.02	- 1.9
6	- 1.32	- 0.05	- 0.39	2.3
8	- 1.09	- 0.31	- 0.40	3.3
2	- 2.68	1.40	1.06	4.4
4	- 1.38	1.19	0.75	5.4
5	- 1.63	1.33	1.02	6.4
3 {test}	- 1.67	1.62	1.82	13.4 {next epoch}
9 {test}	0.28	121	122	158
(1) . . . AVHRR data - old PIXPO,				
(2) . . . AVHRR data - PIXPO 3,				
(3) . . . AVHRR data - PIXPO 4.				

* These elements are used within the programmes PIXPO 3 and 4-

5.2. Validation by means of independent programmes

This type of validation can be called "orbital" because of use of the programmes for direct and inverse referencing; the same GCP's as in Section 5.1 (Table 7) are used, but in principle, arbitrary points (with no relationship to the AVHRR images or similar data) could be used for this goal.

Direct versus inverse referencing. First we compute the geodetic coordinates of a subsatellite point by means of ORBITEST (for given t and mean elements in $t_o \neq t$), and then, we seek this t from those geodetic coordinates (by using the same elements in t_o) with the programmes PIXPO 3 and 4.

Inverse referencing with various programmes. The old PIXPO and the new PIXPO 3 or 4 are independent. It is assumed also that the AVHRR observations and the old PIXPO are independent. PIXPO 3 (modified EPHEM) and PIXPO 4 are independent in the method and they have independent mains of the programmes (while the subroutines are common, but already tested on independent software for the orbit determination). So, we compute the time of culmination $t(l)$ and the off-nadir angle $\delta(p)$ repeatedly, by PIXPO, PIXPO 3 and PIXPO 4, and we compare them to the AVHRR observations (Tables 8 and 9).

We were informed (Rosborough, 1991, private commun.) that a totally independent software for both direct and inverse referencing has recently been

worked out at the University of Colorado, Boulder, USA. A part of their software is able to make an altitude correction for the satellite, based on GCP's (error No. 10, Table 1); this error was out of our scope yet. They also have similar problems with the orbital elements. Till the deadline of this report (end of July 1991), numerical results from Colorado for comparison purposes have not been at our disposal.

5.3. Test of inverse referencing of images

Old and new versions of the indirect referencing routines, PIXPO and PIXPO 3, respectively, have been applied to AVHRR images. The area under consideration is a part of the North Sea and adjacent coastlines and land areas bounded by 50°N, 0°E and 56°N and 10°E. In particular, the coastlines can be identified within the images giving the best mean of comparison for different maps of the same area. If the geolocation of two images is correct, assuming an error of the size not larger than a resolution element of the sounding instrument, then the coastlines within both images should match each other in every point of the image. In this case an image that contains the sum of two geolocated images from different satellite overpasses will show only one coastline. Otherwise, if the geolocation bears larger errors, one should identify duplicated coastlines.

We have chosen two NOAA-9 passages during August 10 and 11 in order to test the geolocation. For the old PIXPO the times and longitudes of the most recent equator crossings are specified with an accuracy of one second and a hundredth of a degree, respectively. For PIXPO 3 we make use of NASA two-line elements given for August, scene No. 3 in Table 8. The images have been mapped to an equidistant grid within above mentioned coordinates resolving 512 x 512 pixels. This leads to a resolution of 1.3 km, corresponding approximately to the pixel size of AVHRR measurements. The resulting images are shown in Fig. 9a and 9b for PIXPO and PIXPO 3, respectively. While the result for PIXPO clearly reveals two coastlines and duplicated East Frisian Islands which are separated by about 12 km in latitude, the result for PIXPO 3 only shows a single coastline which is nowhere smeared by more than one pixel. The accuracy of PIXPO 3 is also demonstrated by the appearance of the Elbe river whose width is narrowed to about three kilometers when going upstream about 30 km from its mouth.

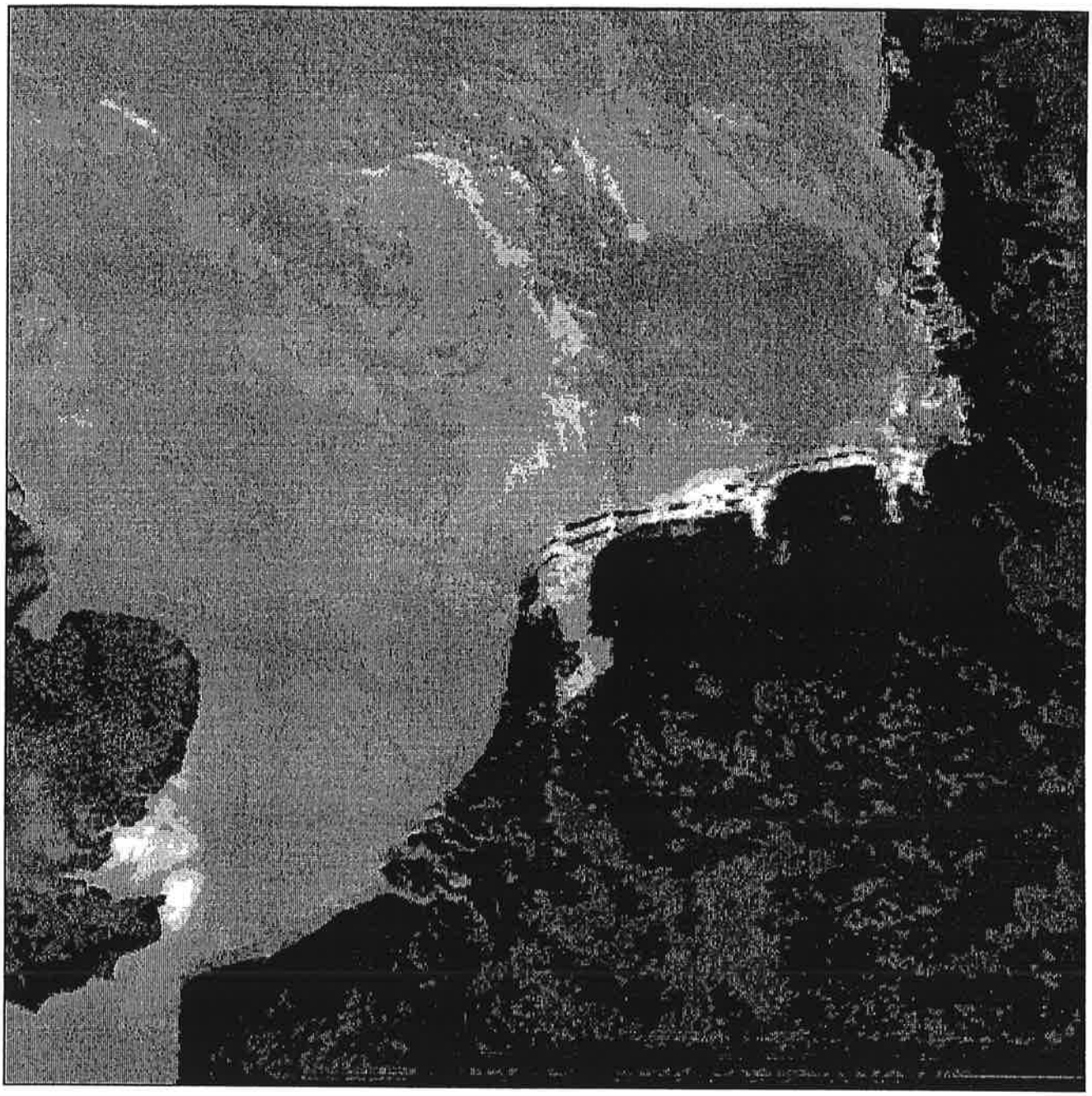


Fig. 9a: Test of inverse referencing of images (NOAA-9). The old PIXPO.

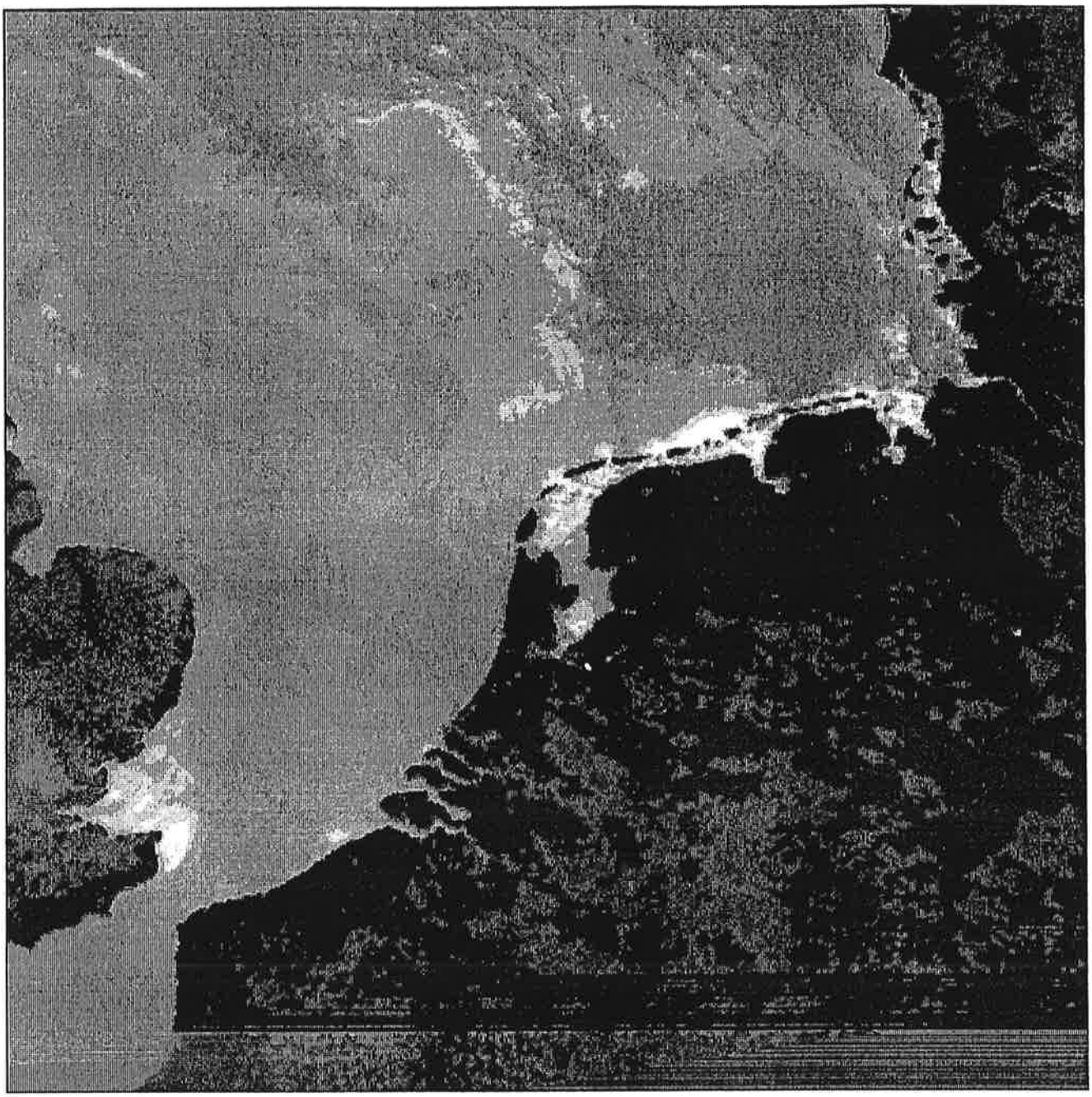


Fig. 9b: Test of inverse referencing of images (NOAA-9). PIXPO 3.

5.4. Interpretation

The results from 6 scenes of NOAA-9 using NASA two-line elements for orbit information are given in Table 8: the AVHRR observations are compared to $t(l)$ and $\delta(p)$ from the old PIXPO programme and to an updated version of EPHEM 5 (called PIXPO 3) and the new PIXPO 4. The differences between the results are usually of the order of 0.5 - 1.5 sec in $t(l)$ and 0.1° in $\delta(p)$ and are mostly of systematic character; only if they are very small they change the sign within one scene. As the variability of the differences within the particular scene is not large, we can define and work with "mean differences per scene", for AVHRR - old PIXPO, AVHRR - PIXPO 3, and AVHRR - PIXPO 4, respectively. Such an information is given in Table. 9.

It is evident from Table 9 that the differences between AVHRR observations and results from the new programmes are much lower than those between the AVHRR data and old PIXPO provided that the orbital elements are only a few days old: compare the scenes 3, 6 and 8 to 3{test} and 9{test}. The differences between the AVHRR data and the old and new programmes are of comparable magnitude for 1-2 weeks old elements, see scenes Nos. 4, 5 and 3{test}. In the extreme case [9{test}], when the elements "were not available" during 158 days, the navigation with PIXPO 3 and 4 cannot be successful; the old PIXPO requires t_{eq}, λ_{eq} of the most recent equatorial crossing - $(t - t_{eq}) \sim 1/2$ hour - and thus, it navigates with an accuracy depending on the accuracy of t_{eq}, λ_{eq}. If t_{eq}, λ_{eq} are not available, the old PIXPO is lost while PIXPO 3 and PIXPO 4 do not require to know t_{eq}, λ_{eq} at all.

The most important conclusions which can be drawn from Table 9 is that the new PIXPO 3 and 4 are potentially more accurate than the old PIXPO. They can compute $t(l)$ with about ~0.2 sec (~2km) and $\delta(p)$ with about ~0.02° (~2km), i.e. roughly with the pixel-size accuracy in total. Further improvement is meaningful provided that more accurate, well documented orbital elements are available; then some improvements of the software, namely of the perturbation model for the PIXPO 3 and 4 are necessary. Our institutions have the relevant parts of the software already available and we are ready to implement them into PIXPO 3 and 4.

Comments:

The previous validation was performed by means of the GCP's and by comparing more or less independent software for the inverse referencing. The comparison "direct versus inverse" referencing - mentioned in Section 5.2 - is missing here due to the following reason.

The programmes ORBITS and ORBITEST are based on the Brouwer-Lyddane theory and work with the TBUS elements. The programmes PIXPO 3 and 4 are based on the Kozai theory for the NASA two-line elements (NASA, 1974) and, as described in Section 2, we were not able to connect one type of the elements to the other because we know nothing about the real perturbation model of the TBUS elements. When ORBITEST and PIXPO 3 and 4 are used to perform the test "direct versus inverse" referencing, the discrepancy in a of the satellite orbit of the order of 4-5 km leads to about 3 sec shift in the time of culmination between TBUS and NASA two-line elements.

6. Conclusion

The software for the direct and inverse referencing (ORBITS, PIXPO) has been studied and improved, or replaced by another software (ORBITEST, PIXPO 3 and 4). The validation tests show that the accuracy achieved in determination of the time of culmination (or pixel line) and in the off-nadir angle (or pixel number) is comparable to the present pixel size for the NOAA-N satellites, i.e. 1-5 km, provided that the orbital elements (NASA two-line elements) are not older than about two weeks. The main obstacle for a further significant improvement is the low quality of the orbital elements.

7. Recommendations

1. To use the TBUS elements properly we need to now their mathematical description at least to the level known for the NASA two-line elements (NASA, 1974). Till now we have at our disposal only NOAA/NESS (1981), and it is not sufficient.
2. It is believed that the NOAA-N orbit observations and the old PIXPO programme are independent, but more information is needed to know the method how the time of culmination and the off-nadir angle for the AVHRR data on board the NOAA-N satellite are computed.

3. A detailed comparison of our results from Table 8 with the results obtained by an independent American software (Rosborough, 1991, priv. commun.) would be very useful.
4. A list of requirements for orbital elements (Section 2), as formulated by remote sensing people, should be presented to the relevant space agencies.
5. The calibration of satellite clocks is needed in order to reduce the "timing error". This information is now available for current satellites from NOAA's Electronic Bulletin Board to be accessed via Omnet/Science net.
6. An inclusion of the attitude correction software to our software is required.
7. The force model of present PIXPO 3 and 4 programmes corresponds to that of the NASA two-line elements, i.e. to the accuracy of satellite position obtainable by them, and to our present pixel size 1 x 1 km^2. A project "Algorithms and validation of ATSR" to investigate sea surface temperatures in situ and from ESA's ERS-1 is under way. The pixel size of the Along Track Scanning Radiometer should be 500 x 500 m^2. The orbital elements will be based on accurate satellite observations by PRARE (Precise Range and Range-Rate Equipment, Germany) and SLR (Satellite Laser Ranging) from a worldwide geodetic network. The orbit determination of ERS-1 will include all known forces affecting the satellite orbit. It is estimated that the along-track accuracy will be between 10 and 100 m and the accuracy in the radial direction will not be worse than 1 m. Our present navigation software ORBITEST, PIXPO 3 and 4 must be improved accordingly. The force model should include "all" zonal and some of the low degree and order tesseral harmonic coefficients (for a rough guide see Table 4 in Straka et al., 1991), a standard atmospheric model and luni-solar perturbations. Subroutines for this purpose are available at Astronomical Institute of the Czech Academy of Sciences.

8. *References*

Cappellari, J.O., C.E. Velez, A.J. Fusch (1976): Mathematical theory of the Goddard trajectory determination system. GSFC Rep. X-582-76-77, Greenbelt.

Colombo, O.L. (1984): Altimetry, orbits and tides. NASA Techn. Memor., 86189, GSFC, Greenbelt.

Decker, B.L. (1986): WGS84. in: Proc. 4th Geodet. Symp. on Satel. Positioning, Texas University, Vol. I, 69-92.

Emery, W.J., J. Brown, Z.P. Nowak (1989): AVHRR image navigation: summary and review. Photogram. Eng. and Remote Sensing, 55, 1175.

Ho, D., A. Asem (1986): NOAA AVHRR image referencing. Int. J. Remote Sensing, 7, 895.

Kozai, Y. (1959): The motion of a close Earth satellite. Astronom. J., 64, 367.

Kostelecky, J. (1990): EPHEM 5 - an improved version of a programme for satellite ephemerides. Documentation at Pecny Observatory. Research Inst. for Geodesy Praha-Zdiby (in Czech, unpubl.).

Lála, P. (1968): A computer programme for computation of ephemerides of artificial Earth satellites. Bull. ITCP Sept. 12, Wahsington, D.C.

Lála, P. (1981): Computer programme PRIOR used for orbit determination at the Ondrejov Observatory. Advances Space Res., 1, 57.

Lundquist, C.A., G. Veis (1966): Geodetic parameters for a 1966 Smithsonian Standard Earth. Vol 1, Smiths. Astrophys. Obs. Spec. Rep., 200, Cambridge, Mass.

Marsh, J.G., and 17 others (1987): A new gravitational model for Earth from satellite tracking data GEM T1. NASA Techn. Memor., 4019, GSFC, Greenbelt.

Marsh, J.G., and 16 others (1989): The GEM T2 gravitational model. NASA Techn. Memo., 100746, GSFC, Greenbelt.

NASA (1974): Explanation of the NASA Prediction Bulletin. NASA Mission Oper. Divis. Report.

NOAA/NESS (1981): NOAA-Telemetry. Revision Nr. 1, TBUS elements description.

Robinson, I.S. (1985): *Satellite Oceanography*. Ed.: Ellis Horwood Lim., England.

Schwintzer, P., Ch. Reigber, G. Balmino, R. Biancale (1990): GRIM-4, a new global Earth gravity model. Presented at 1st Int. Geod Commiss. Symp., Milano, Italy.

Sima, Z., Klokocník, J., P. Hadrava (1983): A comparison of Earth models by means of their higher frequency parts. Bull. Astronom. Insts. Cs., 34, 302.

Smith, E.A. (1980): Orbital mechanics and analytical modelling of meteorological satellite orbits, applications to the satellite navigation problem. Dept. Atmosph. Scio., paper 321, Colorado State Univ., Fort Collins, Colorado.

Straka, J., J. Klokocník, H. Grassl, (1991): Navigation of satellite measurements without ground control points. Submitted to Int. J. of Rem. Sensing.

Vanicek, P., E. Krakiwski (1986): *Geodesy: The Concepts*. North-Holland, Amsterdam.

Appendix 1

The Subroutine GEOGEO for Latitude Transformation

```
      subroutine geogeo (vecpos,vlon,gclat,gclon)
c     ===========================================================
c
c     this routine transforms  geocentric latitude (on spherical Earth)
c     to geodetic latitude (on rotational best-fitting  ellipsoid)
c
c     /parameters of the reference ellipsoid are "da" and "e2";
c      recent internationally accepted values are choosen;
c      if this subroutine is used in conjuction with NASA-two line
c      elements, computed by means of a program, where other values of
c      the constants are used, those other values should replace
c      our values/.
c
      implicit real*8 (a-h, o-z)
      dimension   geo(10), vecpos(3)
      real*4  vlon,clon,gclon,gclat
      parameter (dtr = 6.283185307179586d0/360.d0, rtd=1.d0/dtr)
      arctan(x,y) = rtd*datan2(y,x)
c
      da=6378.137d0
      e2=0.0066944d0
      epsilon=1d-4
c     this is the limit for accuracy of iteration from geocentric
c     to geodetic latitude, epsilon is in degrees.
c
      clon = arctan (vecpos(1), vecpos(2))
      deqr=dsqrt(vecpos(1)*vecpos(1) + vecpos(2)*vecpos(2))
      gclat = arctan(deqr,vecpos(3))
      gclon = clon + vlon
      if(gclon .lt. -180.) gclon = gclon + 360.
      if(gclon .gt.  180.) gclon = gclon - 360.
c
c     write(6,329) clon,deqr,gclat,gclon
c 329 format(4d13.6)
c
      if (gclat .eq. 90.)   gclat= 89.99d0
      if (gclat .eq.(-90.)) gclat=-89.99d0
c     this is a provisional escape from singularity at poles
c
      gclar = gclat *dtr
      dsfi0 = dsin(gclar)
      dcfi0 = dcos(gclar)
      dn0 = da/ dsqrt(1d0-e2*dsfi0*dsfi0)
      dh0 = (deqr/dcfi0) - dn0
c     dh0 = 0d0
c     the iteration may start with latter as well as former "dh0"
      dhn0 = dh0 + dn0
      ni = 0
      darg1 = dhn0/(dhn0-e2*dn0)
      vecpo1 = vecpos(3)*darg1

      geolat1 = arctan (deqr,vecpo1)
      dif0= geolat1-gclat
c     write(6,330) ni,geolat1,gclat,dif0,dh0,dn0,darg1
c 330 format(i4, 6d13.6)
c
      if(dabs(dif0).lt.epsilon) go to 1000
c     user can change this limit "epsilon" if less or more
c     precise iteration is needed; present limit is compatible
c     with a 100meter accuracy in (phi,lambda).
c
      ni = 2
      geo(1)=0d0
      geolat=geolat1
    1 nim1=ni-1
      geo(nim1)=geolat
      geolar=geolat * dtr

      dsfi = dsin (geolar)
      dcfi = dcos (geolar)
      dn = da / dsqrt(1d0-e2*dsfi*dsfi)
      dh = (deqr/dcfi) - dn
      dhn = dh +dn
      darg = dhn/(dhn-e2*dn)
      vecpoi=vecpos(3)*darg
      geolat = arctan (deqr,vecpoi)
      geo(ni)=geolat
      dif = geo(ni) - geo(nim1)
c     write(6,330) nim1,geo(ni),geo(nim1),dif,dh,dn,darg
      if (dabs(dif).lt.epsilon) go to 1000
      ni = ni + 1
      go to 1
 1000 continue
c
c     rename variables for output:
      if (ni.eq.0) gclat=geolat1
      if (ni.gt.1) gclat=geo(ni)
c
      return
      end
```

Appendix 2

Coordinates of the Pole and Universal Time in 1986

COORDINATES OF THE POLE AND UNIVERSAL TIME AT 5-DAY INTERVAL IN 1986

COMBINATION OF ALL AVAILABLE DATA
UNITS ARE, FOR X,Y, 0.0001", FOR UT, 0.00001S

CALEND. (OH UT)	MJD	BESSEL. YEAR 1986.	X	Y	UT2-UTC	UT1-UTC	STAND. ERR. X	Y	UT
1986 JAN 4	46434	9	1810	1643	30324	30769	9	8	5
1986 JAN 9	46439	23	1704	1555	29732	30103	6	5	4
1986 JAN 14	46444	37	1611	1434	29181	29486	8	7	4
1986 JAN 19	46449	50	1512	1365	28394	28640	7	7	4
1986 JAN 24	46454	64	1365	1312	27898	28088	7	7	4
1986 JAN 29	46459	78	1231	1267	27262	27395	6	5	3
1986 FEB 3	46464	91	1074	1213	26281	26353	9	8	4
1986 FEB 8	46469	105	953	1199	25686	25693	8	8	4
1986 FEB 13	46474	119	836	1146	24856	24787	8	7	4
1986 FEB 18	46479	133	713	1146	24183	24026	8	8	6
1986 FEB 23	46484	146	576	1147	23848	23591	8	9	5
1986 FEB 28	46489	160	400	1153	23121	22748	7	7	4
1986 MAR 5	46494	174	266	1159	22536	22032	7	7	4
1986 MAR 10	46499	187	111	1183	22049	21398	9	9	4
1986 MAR 15	46504	201	-46	1204	21419	20605	9	8	5
1986 MAR 20	46509	215	-139	1217	21267	20276	8	8	4
1986 MAR 25	46514	228	-238	1304	20870	19688	9	8	4
1986 MAR 30	46519	242	-383	1345	20039	18657	9	9	5
1986 APR 4	46524	256	-487	1414	19605	18018	7	7	4
1986 APR 9	46529	269	-591	1493	18889	17094	7	6	3
1986 APR 14	46534	283	-645	1612	18318	16315	9	9	5
1986 APR 19	46539	297	-683	1737	17890	15686	12	10	5
1986 APR 24	46544	310	-778	1817	17061	14668	7	8	4
1986 APR 29	46549	324	-819	1900	16444	13874	7	7	4
1986 MAY 4	46554	338	-933	2036	15806	13082	13	13	8
1986 MAY 9	46559	352	-910	2115	14940	12087	8	8	4
1986 MAY 14	46564	365	-930	2190	14443	11490	8	8	4
1986 MAY 19	46569	379	-975	2333	13895	10875	13	12	6
1986 MAY 24	46574	393	-985	2406	13093	10043	11	10	5
1986 MAY 29	46579	406	-1013	2530	12658	9616	8	8	4
1986 JUN 3	46584	420	-1007	2617	12007	9014	9	9	5
1986 JUN 8	46589	434	-1002	2753	11633	8730	8	8	4
1986 JUN 13	46594	447	-995	2846	11428	8658	10	9	5
1986 JUN 18	46599	461	-916	2933	10747	8150	8	8	4
1986 JUN 23	46604	475	-834	3036	10162	7777	9	9	5
1986 JUN 28	46609	488	-758	3174	9505	7368	9	10	5
1986 JUL 3	46614	502	-681	3274	8763	6908	9	9	5
1986 JUL 8	46619	516	-637	3348	8303	6761	8	7	4
1986 JUL 13	46624	530	-570	3471	7609	6401	8	7	4
1986 JUL 18	46629	543	-476	3535	6807	5952	9	8	5

CALEND. (OH UT)	MJD	BESSEL. YEAR 1986.	X	Y	UT2-UTC	UT1-UTC	STAND. ERR. X	Y	UT
1986 JUL 23	46634	557	-353	3607	6233	5743	11	12	6
1986 JUL 28	46639	571	-319	3637	5355	5237	9	8	5
1986 AUG 2	46644	584	-241	3693	4818	5073	7	7	4
1986 AUG 7	46649	598	-161	3763	4349	4970	9	8	5
1986 AUG 12	46654	612	-118	3791	3398	4377	9	10	5
1986 AUG 17	46659	625	-35	3804	2765	4081	10	9	5
1986 AUG 22	46664	639	75	3879	2031	3661	12	10	6
1986 AUG 27	46669	653	167	3914	1267	3183	10	8	4
1986 SEP 1	46674	666	239	3947	868	3038	11	11	6
1986 SEP 6	46679	680	287	3991	179	2568	9	7	5
1986 SEP 11	46684	694	382	3958	-660	1911	11	9	6
1986 SEP 16	46689	708	470	3999	-1232	1482	11	8	4
1986 SEP 21	46694	721	498	3948	-2215	600	12	10	6
1986 SEP 26	46699	735	568	3946	-2869	8	12	10	6
1986 OCT 1	46704	749	605	3855	-3518	-619	17	12	10
1986 OCT 6	46709	762	686	3927	-4570	-1684	10	10	6
1986 OCT 11	46714	776	740	3888	-5213	-2376	12	10	6
1986 OCT 16	46719	790	796	3890	-5985	-3228	7	6	3
1986 OCT 21	46724	803	912	3877	-6757	-4109	11	10	6
1986 OCT 26	46729	817	935	3819	-7211	-4694	13	11	5
1986 OCT 31	46734	831	997	3864	-7950	-5584	11	11	6
1986 NOV 5	46739	844	1053	3808	-8629	-6427	15	13	6
1986 NOV 10	46744	858	1154	3765	-9016	-6989	14	12	6
1986 NOV 15	46749	872	1206	3713	-9736	-7888	13	11	6
1986 NOV 20	46754	885	1220	3642	-10109	-8442	12	10	6
1986 NOV 25	46759	899	1239	3580	-10459	-8971	10	10	5
1986 NOV 30	46764	913	1262	3518	-11199	-9883	9	9	5
1986 DEC 5	46769	927	1312	3454	-11580	-10426	12	10	6
1986 DEC 10	46774	940	1394	3392	-12170	-11167	10	8	4
1986 DEC 15	46779	954	1414	3305	-12653	-11789	10	10	5
1986 DEC 20	46784	968	1409	3249	-12881	-12140	12	11	6
1986 DEC 25	46789	981	1457	3213	-13546	-12917	15	14	7
1986 DEC 30	46794	995	1476	3143	-14185	-13653	10	9	5

UT2 - UT1

The following conventional formula is used :

UT2-UT1 = $0.0220 \sin 2\pi t - 0.0120 \pi \cos 2\pi t - 0.0060 \sin 4\pi t + 0.0070 \cos 4\pi t$, the unit being the second and t being the date in besselian years.

COORDINATES OF THE POLE AND UNIVERSAL TIME
AT 5-DAY INTERVAL IN 1987

COMBINATION OF ALL AVAILABLE DATA
UNITS ARE, FOR X,Y, 0.0001", FOR UT, 0.00001S

CALEND. (OH UT)	MJD	BESSEL. YEAR 1987.0	X	Y	UT2-UTC	UT1-UTC	STAND. ERR. X	Y	UT
1987 JAN 4	46799	9	1474	3101	-14728	-14280	6	6	5
1987 JAN 9	46804	22	1462	3037	-15506	-15132	6	6	4
1987 JAN 14	46809	36	1404	2969	-15831	-15523	7	5	5
1987 JAN 19	46814	50	1415	2883	-16264	-16015	5	4	4
1987 JAN 24	46819	63	1366	2814	-16998	-16805	6	6	5
1987 JAN 29	46824	77	1369	2776	-17416	-17281	4	4	4
1987 FEB 3	46829	91	1310	2715	-18214	-18139	5	4	3
1987 FEB 8	46834	105	1304	2645	-18857	-18848	5	5	4
1987 FEB 13	46839	118	1270	2607	-19300	-19365	6	6	5
1987 FEB 18	46844	132	1267	2562	-20115	-20267	4	4	3
1987 FEB 23	46849	146	1242	2519	-20836	-21088	4	4	4
1987 FEB 28	46854	159	1220	2451	-21623	-21989	6	5	5
1987 MAR 5	46859	173	1193	2389	-22726	-23224	8	8	6
1987 MAR 10	46864	187	1120	2300	-23254	-23898	6	6	5
1987 MAR 15	46869	200	1117	2252	-23870	-24677	6	6	5
1987 MAR 20	46874	214	1096	2227	-24735	-25718	5	5	5
1987 MAR 25	46879	228	1019	2192	-25298	-26470	4	4	3
1987 MAR 30	46884	241	970	2162	-26145	-27516	6	6	5
1987 APR 4	46889	255	952	2141	-26758	-28334	5	5	4
1987 APR 9	46894	269	895	2119	-27187	-28973	4	5	3
1987 APR 14	46899	283	822	2090	-27961	-29954	6	6	5
1987 APR 19	46904	296	735	2057	-28529	-30723	7	7	5
1987 APR 24	46909	310	709	2039	-29095	-31480	4	5	4
1987 APR 29	46914	324	659	2049	-29917	-32477	6	6	6
1987 MAY 4	46919	337	584	2039	-30320	-33036	4	4	3
1987 MAY 9	46924	351	552	2004	-30861	-33708	4	4	4
1987 MAY 14	46929	365	480	2004	-31618	-34567	8	8	6
1987 MAY 19	46934	378	399	1988	-32071	-35089	6	6	5
1987 MAY 24	46939	392	351	1996	-32914	-35963	6	6	6
1987 MAY 29	46944	406	298	1994	-33593	-36636	7	6	6
1987 JUN 3	46949	419	272	1990	-34058	-37055	6	6	6
1987 JUN 8	46954	433	241	2002	-34840	-37748	6	6	4
1987 JUN 13	46959	447	189	2005	-35479	-38257	5	5	5
1987 JUN 18	46964	461	128	2019	-36214	-38819	5	5	4
1987 JUN 23	46969	474	68	2042	-37039	-39434	4	4	3
1987 JUN 28	46974	488	-20	2063	-37439	-39588	7	7	6
1987 JUL 3	46979	502	-50	2104	-38057	-39925	8	7	6
1987 JUL 8	46984	515	-108	2113	-38827	-40387	6	6	5
1987 JUL 13	46989	529	-176	2140	-39355	-40582	7	7	6
1987 JUL 18	46994	543	-230	2177	-40253	-41127	6	6	4

CALEND.	MJD	BESSEL. YEAR 1987.	X	Y	UT2-UTC	UT1-UTC	STAND. ERR. X	Y	UT
1987 JUL 23	46999	556	-248	2213	-40769	-41275	5	5	4
1987 JUL 28	47004	570	-245	2270	-41218	-41353	8	7	7
1987 AUG 2	47009	584	-269	2293	-41985	-41747	5	5	4
1987 AUG 7	47014	597	-305	2344	-42535	-41929	6	6	5
1987 AUG 12	47019	611	-319	2374	-43315	-42354	5	5	4
1987 AUG 17	47024	625	-360	2430	-44132	-42834	8	8	6
1987 AUG 22	47029	638	-378	2493	-44633	-43019	9	9	6
1987 AUG 27	47034	652	-373	2574	-45504	-43601	6	6	4
1987 SEP 1	47039	666	-365	2620	-46411	-44252	7	7	6
1987 SEP 6	47044	679	-393	2669	-47039	-44659	7	7	6
1987 SEP 11	47049	693	-404	2698	-48125	-45562	8	8	6
1987 SEP 16	47054	707	-433	2767	-48771	-46064	9	9	6
1987 SEP 21	47059	721	-474	2819	-49404	-46593	8	7	6
1987 SEP 26	47064	734	-508	2889	-50418	-47543	6	5	4
1987 OCT 1	47069	748	-556	2965	-51089	-48189	7	7	6
1987 OCT 6	47074	762	-555	3024	-51964	-49077	7	7	7
1987 OCT 11	47079	775	-589	3083	-52897	-50057	6	6	5
1987 OCT 16	47084	789	-629	3150	-53394	-50633	5	5	3
1987 OCT 21	47089	803	-625	3216	-54171	-51516	6	6	5
1987 OCT 26	47094	816	-625	3268	-54933	-52409	6	6	5
1987 OCT 31	47099	830	-617	3342	-55474	-53099	6	6	5
1987 NOV 5	47104	844	-608	3399	-56403	-54194	6	6	5
1987 NOV 10	47109	857	-551	3464	-56956	-54921	8	7	6
1987 NOV 15	47114	871	-565	3569	-57551	-55695	11	11	7
1987 NOV 20	47119	885	-540	3638	-58380	-56705	11	11	8
1987 NOV 25	47124	898	-545	3713	-58928	-57432	7	6	5
1987 NOV 30	47129	912	-599	3788	-59714	-58389	6	6	5
1987 DEC 5	47134	926	-589	3843	-60457	-59295	6	6	5
1987 DEC 10	47139	940	-524	3907	-60871	-59862	6	5	4
1987 DEC 15	47144	953	-473	3992	-61548	-60677	6	5	5
1987 DEC 20	47149	967	-410	4044	-62225	-61479	8	8	10
1987 DEC 25	47154	981	-358	4079	-62859	-62225	6	7	5
1987 DEC 30	47159	994	-283	4119	-63836	-63299	10	10	7

UT2 - UT1

The following conventional formula is used :

$UT2-UT1 = 0.0220 \sin 2\pi t - 0.0120 \cos 2\pi t - 0.0060 \sin 4\pi t + 0.0070 \cos 4\pi t$, the unit being the second and t being the date in besselian years.

Appendix 3

Programme EPHEM 7 Calculating Ephemerides

```
c     program ephem7 - including new sao format for longperiodic
c     perturbations, precise computation of the true sidereal time and
c     different perturbation effects - prepared by p.lala, june 1987
c     disc copy corrected (itype) october 1988
c     pc version: j. kostelecky, july 91
c     subroutines used: ahms,angle,antk,calday,covel2,deltae,elas,elemen
c                       elsol,epler,inel,jayday,kozai,krizak,outel,phase
c                       posic2,sapla,sbg1,statn1,topo,vect,costa,nhms,sh
c
      implicit real*8(a-h,o-z)
      parameter (idim=16)
      integer prv(idim),pos(idim)
      character*8 sate
      character ff
      character*80 jmeno,radek
      dimension tm(90),az(90),he(90),ds(90),ar(90),de(90),
     1rt(90),dr(90),sm(90),ud(90),lt(90),faze(90),sol(250)
      dimension xst(12),xv(6),nsta(16),na(8)
      dimension p(15),zst(12, 9),nstb( 9),nstc(16),pp(15)
      dimension en(250),el(250),gl(250),abl(250,9) ,ihs(90,9)
      common/el/ al(100,8),dal(100,6)
      common/type/itype
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     1nl,jp1,jg1,jil,je1,ja1,jm1
      data tx1,tx2,tx3/8h day    ,8h night  ,8h        /
      data na/4h wed,4h thu,4h fri,4h sat,4h sun,4h mon,4h tue,4h    /
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
      data  dhg,hg1/7.2921151463d-5,6.300387487d0/
      data  rz,dmi/6.37814d3,398600.5d0/
      data vl/299792.458d0/,fb/6.73952453d-3/
      do 9876 ijkl=1,100
      do 9876 ijklm=1,6
 9876 dal(ijkl,ijklm)=0d0
      do 9875 ijkl=1,16
 9875 nsta(ijkl)=0
      ff=char(12)
      eps=23.4523d0*cdr
      se=dsin(eps)
      ce=dcos(eps)
      itype=55
      print *,'program  e p h e m  for calculation of satellite culmination
     &s'
      print *,'give the name of input file, (this file must be *.dat)'
      read(*,'(a)') jmeno
      ijm=index(jmeno,' ')-1
      write(*,7001) jmeno(:ijm)
 7001 format(' output will be in the file:',a,'.lst')
      open(55,file=jmeno(:ijm)//'.dat',status='unknown')
      open(66,file=jmeno(:ijm)//'.lst',status='unknown')
c
c     read control card
c
 1000 read(55,*,end=555)lp,lpj3,tskew
    1 format(i3,2f5.1,2i3)
      if (lp.ne.-5) n5=0
      nn=iabs(n5)
   71 ilem=iabs(lp)
      if(krok.le.0) krok=30
c
c     read orbital elements (and amplitudes of lgp perturbations)
c
      call inel(ilem)
      print *,'inel'
      if(lp)77,77,83
   77 read(itype,79 ) p(1),pp(4), pp(7), p(10), p(13),p(2),pp(5),
     1 pp(8), p(11),p(14), p(3),pp(6),pp( 9), p(12),p(15),pp(1)
   79 format(10x,5d14.7)
      do 86 i=4,9
   86 p(i)=pp(i)*cdr
c
c     read time
c
   83 read(55,*)jnl1,a,b,c,aut,but,jdu
    3 format(2(i6,2f3.0,f7.3),1x,2f10.2,i5)
      print *,'cas'
      if(jnl1.le.99999)goto 100
      ny=jnl1/10000
      i=jnl1-ny*10000
      nm=i/100
      nd=i-nm*100
      ny=ny+1900
      call jayday(nd,nm,ny,jnl1)
  100 ajnl1=jnl1+(a*36d2+b*6d1+c)/864d2
      enlb=ajnl1-djne
      djds=djne
      cut=(aut+but*(djds-jdu))/864d2
c     call nuta83(djds)
      hg0=dthg83(djds+cut)
      call outel(ilem)
      if(lpj3.ne.0) call longj3(enlb)
      cr=dcos(ei(1))
      sr=dsin(ei(1))
      kb=1
      ksta=25000
      print *,'outel'
      if (lp.lt.0) write(66,89)(p(i),i=1,3),(pp(i),i=4,9),
     *(p(i),i=10,15),pp(1)
   89 format(' ',71h  amplitudes of long-periodic perturbations are take
     *n from input data :/(1x,3d16.7))
      call ahms(nh,nm,s,hg0/dpi)
      write(66,60) djne,nh,nm,s,aut,but,jdu,tskew
   60 format(' ',18h  sidereal time at,f8.1,3h : ,2(i2,1h:),f6.3/
     *3x,'correction (ut1 - utc)   : ',2(1x,f10.5),i6/'   tskew (sec):'
     #,f7.3/)
c cycle for different points
      write(66,301)
  301 format(' point no. fi(geod)    lambda    h(m)   date   '
     &,'   time culm.  delta(deg)'/)
      do 23 ista=1,ksta
      img=55
      call costa2(img,nst,xs,ys,zs,sl,sf,ht)
      print *,'sour',xs,ys,zs,sl,sf,ht
   64 a=sl/cdr
      b=sf/cdr
      c=ht*1d6
c     write(66,6)nst,xs,ys,zs,a,b,c
    6 format(' prediction is made for point no.',i5,
     1' with coordinates:'/2x,3f10.6,2f11.6,f7.1)
      xs=xs*1d3
      ys=ys*1d3
      zs=zs*1d3
      cfg=dcos(sf)
      sfg=dsin(sf)
      gg= dsqrt(xs*xs+ys*ys)
      rs= dsqrt(xs*xs+ys*ys+zs*zs)
      gs=datan(zs/gg)
      xst(1)=xs
      xst(2)=ys
```

```
      xst(3)=zs
      xst(4)=rs
      xst(5)=gg
      xst(6)=sl
      xst(7)=gs
      xst(8)=sf
      xst(9)=sfg
      xst(10)=cfg
      xst(11)=ht
      cfg=dcos(gs)
      sfg=dsin(gs)
c
c     compute parameters of stations for simultanity test
c
   76 kstb=k
c
c * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c
c part 1: computation of parameters at culmination
c
c * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c
      a=dpi*25d-2
      anrv=ep(1)-a
      dl=eg(1)-a
      aa=ea(1)
      ae=ee(1)
      c=1d0
      n=0
      k=0
      i1=0
      ndate=0
      djdm=0d0
      enl=enlb
c
   53 l=0
      m=0
   52 gv=aa*(1d0+ae)
      if(kb.eq.0) m=1
   61 ag=dl+eg(2)*enl
      hg=hg0+hg1*enl
      rno=sl+hg-ag
      a=dsin(rno)
      crgo=sfg*cr+cfg*sr*dcos(rno)
      b=1d0-crgo*crgo
c
c      if (b.lt.1d-2) goto 24
      print * ,'61 ag',ag,crgo,b
   18 srgo=dsqrt(b)
      snro=-cfg*a/srgo
      b=dmax1(1d-20,1d0-snro*snro)
      cnro=dsqrt(b)
      if(cr*crgo.gt.sfg)cnro=-cnro
      snor=sr*a/srgo
      b=dmax1(1d-20,1d0-snor*snor)
      cnor=dsqrt(b)
      if(sfg*crgo.gt.cr)cnor=-cnor
   19 anro=datan2(snro,cnro)
c    21 if(l+m.eq.0.or.i1.eq.0) go to 72
c
c     rough computation of the time of culmination
c
   72 hk=datan(a/c)/cdr
      anrp=anrv+ep(2)*enl
      a=0d0
      prm=elemen(em,enl,jm1)
      prm=dmod(prm,1d0)*dpi
      if(ilem.eq.3)a=antk( anrp+dpi*25d-2,ae)
      prm=prm-a
      if(m.gt.0) go to 99
      amro=angle(anro-prm-anrp,-dpi)
      if(dabs(amro).lt.9d-1)go to 99
      l=l+1
      enl=enl+amro/(em(2)*dpi+ep(2)-(dpi-eg(2))*cfg*cnor/srgo)
      print *,'enl',enl
      go to 61
   99 pro=angle(anro-anrp,dpi)
      aa=ea(1)+ea(2)*enl
      ae=ee(1)+ee(2)*enl
      if(ae.lt.0d0)ae=0d0
      prmd=antk(pro,ae)
      a=dcos(pro)
      gv=aa*(1d0-ae*ae)/(1d0+ae*a)
      b=angle(prmd-prm,-dpi)
c      if(kb.eq.1) goto 25
c
c     precise computation of the time of culmination
c
   25 dprm=(1d0-ae*a)**2d0/dsqrt((1d0+ae)*(1d0-ae))
      dmdt=em(2)*dpi+ep(2)-dprm*(dpi-eg(2))*cfg*cnor/srgo
      if(dabs(b).le.dmdt*1d-5)goto 120
      m=m+1
      enl=enl+b/dmdt
      print *,'dprm,dmdt,enl',dprm,dmdt,enl
      go to 61
c  120 if(i1.ne.0)go to 121
  120 n=n+1
c time of culmination
      djd=enl+djne
      print *,'djd',djd
      en(n)=enl
c
c     evaluate longperiodic perturbations and prepare for other perturba
c     tion
      if(lp.gt.0) goto 91
      al(n,1)=elemen(ep,en(n),jp1)
      al(n,2)=elemen(eg,en(n),jg1)
      al(n,3)=elemen(ei,en(n),ji1)
      al(n,4)=elemen(ee,en(n),je1)
      a      =elemen(em,en(n),jm1)
      al(n,5)=dmod(a ,1d0)*dpi
      al(n,6)=elemen(ea,en(n),ja1)*1d-3
      al(n,7)=hg
      al(n,8)=(em(2)+2d0*em(3)*en(n))*dpi
      a=al(n,1)
      b=2d0*a
      c=3d0*a
      sp1=dsin(a)
      cp1=dcos(a)
      sp2=dsin(b)
      cp2=dcos(b)
      sp3=dsin(c)
      cp3=dcos(c)
      d=al(n,4)*cp1
      e=al(n,4)*sp1
      f=p(1)*cp1+p(2)*sp2+p(3)*cp3
      g=p(10)*sp1+p(11)*cp2+p(12)*sp3+pp(1)
      d=d+f
```

```
      e=e+g
      h=datan2(e,d)-a
      dal(n,1)=angle(h,-dpi)
      dal(n,4)=dsqrt(d*d+e*e)-al(n,4)
      dal(n,5)=p(13)*cp1+p(14)*sp2+p(15)*cp3
      dal(n,5)=dal(n,5)-dal(n,1)
      dal(n,2)=p(4)*cp1+p(5)*sp2+p(6)*cp3
      dal(n,3)=p(7)*sp1+p(8)*cp2+p(9)*sp3
      dal(n,6)=0d0
c     write(6,78) (dal(n,i),i=1,5)
   78 format(1h ,10x,5d14.7)
      do 90 i=1,5
   90 al(n,i)=al(n,i)+dal(n,i)
c
c     evaluation of the light conditions at the station (or stations)
c
   91 continue
c
c     print of the culmination parameters
c
      djd=djd-tskew/86400d0
      call calday(nd,mm,ny,idint(djd))
      call ahms(nh,nm,s,dmod(djd,1d0))
      kstb=0
c      if(kstb.le.0)
c     &write(66,14) n,en(n),nd,mm,ny,nh,nm,s
c      if(kstb.le.0)
c     &write(*,14) n,en(n),nd,mm,ny,nh,nm,s
c   14 format(' ', 1x,i3,f9.4,4x,2(i2,1h.),i4,1x,2(i2,1h:),f4.1,f5.1,
c     *f6.1,a8,9a6)
c     if(kstb.gt.0)write(66,50)(abl(n,i),i=1,kstb)
c   50 format(1h+,66x,9a6)
c      n=n+1
c  computation of delta
       call posic2(en(n),-1,0,n,xv,u,n3)
      hg=dthg83(djd+cut)
      shg=dsin(hg)
      chg=dcos(hg)
      x1= xv(1)*chg+xv(2)*shg
      y1=-xv(1)*shg+xv(2)*chg
      z1=xv(3)
      print *,'x1,y1,z1', x1,y1,z1
       z1=3d0/86400d0
       call posic2(en(n)+z1,-1,0,n,xv,u,n3)
      hg=dthg83(djd+z1+cut)
      shg=dsin(hg)
      chg=dcos(hg)
      x2= xv(1)*chg+xv(2)*shg
      y2=-xv(1)*shg+xv(2)*chg
      z2=xv(3)
c
      xpo=x1-xs
      ypo=y1-ys
      zpo=z1-zs
      xd1=y1*zpo-z1*ypo
      yd1=z1*xpo-x1*zpo
      zd1=x1*ypo-y1*xpo
      sd=sqrt(xd1**2+yd1**2+zd1**2)
      cd=x1*xpo+y1*ypo+z1*zpo
      delta=atan2(sd,cd)
c sign of delta
      xmax1=dmax1(xd1,yd1,zd1)
      if(dabs(xmax1-xd1).lt.1d-10.and.xmax1/(x2-x1).lt.0d0)
     & then
         delta=-delta
      elseif(dabs(xmax1-yd1).lt.1d-10.and.xmax1/(y2-y1).lt.0d0)
     & then
         delta=-delta
      elseif(dabs(xmax1-zd1).lt.1d-10.and.xmax1/(z2-z1).lt.0d0)
     & then
         delta=-delta
      end if
        delta=-delta
       write(66,14) nst,sf/cdr,sl/cdr,ht*1d6,nd,mm,ny,nh,nm,s
     $,delta/cdr
       write(*,14) nst,sf/cdr,sl/cdr,ht*1d6,nd,mm,ny,nh,nm,s
     $,delta/cdr
   14 format(' ',i6,1x,2f11.6,f6.1,2x,2(i2,1h.),i4,1x,2(i2,1h:),
     @f5.2,f8.2)
   23 continue
      stop
c     if(nn.gt.0) call koste1(npass,nn,nn,0,ep(2),eg(2))
c      adt=0.05
c     if (n5.gt.0) call lsol1(npass,adt,djdp)
  555 stop
      end
c
      subroutine costa2(img,nsta,xs,ys,zs,sa,fg,ht)
      implicit real*8 (a-h,o-z)
      data rz,fz/6.378137d0,0.006694407406d0/,cdr/0.01745329252d0/
      read(img,*,end=1) nsta,  fg,sa, h
      sa=sa*cdr
      fg=fg*cdr
      ht=ht*1d-6
      sf=dsin(fg)
      cv=rz/dsqrt(1d0-fz*sf*sf)
      sv=cv*(1d0-fz)
      cp=(cv+ht)*dcos(fg)
      xs=cp*dcos(sa)
      ys=cp*dsin(sa)
      zs=(sv+ht)*sf
      return
    1 stop
      end
      subroutine ahms(nh,nmin,s,c)
c
c     improved version
c     transformation into hours,minutes and seconds
c    (or into degrees etc. if c is multiplied by 15)
c        input:  c  - fractions of day
c        output: nh - hours
c                nm - minutes
c                s  - seconds (exact to miliseconds)
c
      implicit real*8 (a-h,o-z)
      real*8 s,c
      t=dabs(c*24d0)
      nh=idint(t)
      t=dmod(t,1d0)*60d0
      nmin=idint(t)
      s=dmod(t,1d0)*60d0
      if(dabs(s-6d1).ge.55d-5)go to 1
      nmin=nmin+1
      s=1d-5
      if(nmin.ne.60) go to 1
      nh=nh+1
      nmin=0
    1 if(c.lt.0d0) nh=-nh
```

```
      return
      end
      real function angle*8 (b,r)
      implicit real*8(a-h,o-z)
      real*8 b,r
c
c     normalisation of  angle b into interval:
c     (0,h) if r.gt.0 (where h=dabs(r)), or
c     (-h/2,+h/2) if r.lt.0
c
      h=dabs(r)
      a=dmod(b,h)
      if(r)1,1,2
    1 if(dabs(a).gt.0.5d0*h)a=a-dsign(h,a)
c instrukci go to 3 prehodil j.kost.
      go to 3
    2 if (a.lt.0d0) a=a+h
c     goto 3
    3 angle=a
      return
      end
      real function antk*8(v,e)
c
c     computation of the mean anomaly(rad) from:
c     input:  v-true anomaly(rad)
c             e-eccentricity(-)
c
      implicit real*8 (a-h,o-z)
      real*8 v,e
      a=(1d0-e)/(1d0+e)
      b=v
      if(dabs(dabs(b)-3.14159265) .le.1d-5) goto 1
      f=2d0*datan(dsqrt(a)*dtan(0.5d0*v))
      b=f-e*dsin(f)
    1 antk=b
      return
      end
      subroutine calday(nd,nm,ny,jnl)
c     gives gregorian calendar day,month and year cor-
c     responding to jnl=days since nov.17.0 ut 1858
c
      if(jnl.lt.15384)go to 10
      if(jnl.lt.88067)go to 11
   10 nm=0
      nd=0
      ny=0
      go to 13
   11 nseq=jnl-15078
      ny=(4*nseq-1)/1461
      nd=(4*nseq+3-1461*ny)/4
      nm=(5*nd-3)/153
      nd=(5*nd+2-153*nm)/5
      if(nm.ge.10)go to 12
      nm=nm+3
      ny=ny+1900
      go to 13
   12 nm=nm-9
      ny=ny+1901
   13 return
      end
      subroutine costa1(img,iden,nsta,xs,ys,zs,sa,fg,ht)
c
c     evaluate station coordinates from parameters punched at the
c     card (station is not given in the catalog 'statio')
c   improved 1984 to use input media (img)
c     input : iden- 5 optical station with rectangular coord.
c                   1 laser   station with rectangular coord.
c                  15 optical station with angular coord.
c                  11  laser  station with angular coord.
c     output : nsta-identification number of the station.
c              xs,ys,zs-rectangular coordinates (nm)
c              sa-east longitude (rad)
c              fg-geodetic latitude(rad)
c              ht-height above reference ellipsoid(mm)
c
      implicit  real*8(a-h,o-z)
      data rz,fz/6.37814d0,0.006694407406d0/,cdr/0.01745329252d0/
      read(img,*) nsta,xs,ys,zs,sa,fg,ht,htt
    2 format(i5,3f10.6,f12.6,f12.6,f4.0,f17.1)
      if(iden.ge.10)go to 8
      cv=xs*xs+ys*ys+zs*zs
      cv=dsqrt(cv)/rz
      gg=dsqrt(xs*xs+ys*ys)
      sa=datan2(ys,xs)
      sv=datan2(zs,gg)
      fg=sv
    1 ht=fg
      fg=dsin(ht)
      cr=1d0/dsqrt(1d0-fz*fg*fg)
      cp=fz*cr*fg*dcos(ht)/cv
      cp=dasin(cp)
      fg=sv+cp
      if(dabs(fg-ht).gt.1d- 9) goto 1
      ht=cv*dcos(cp)-1d0/cr
      ht=ht*rz
      goto 7
    8 sa=sa*cdr
      fg=fg*cdr
      ht=ht*1d-6
      sf=dsin(fg)
      cv=rz/dsqrt(1d0-fz*sf*sf)
      sv=cv*(1d0-fz)
      cp=(cv+ht)*dcos(fg)
      xs=cp*dcos(sa)
      ys=cp*dsin(sa)
      zs=(sv+ht)*sf
    7 continue
      return
      end
       real function change*8 (x,dy,de,t)
c
c  computation of the real rate of change of an angle
c  input : x - time difference
c          dy- angle difference
c          de- theoretical rate
c          t - angle unit used
c
      implicit real*8 (a-h,o-z)
      real*8 x,dy,de,t
      dx=dabs(x*de)/t
      j=idint(dx)
      dx=dmod(dx,1d0)
      se=dsign(1d0,de)
      if (se.ne.dsign(1d0,x)) dy=-dy
      dy=angle(dy,t)/t
      dxy=dx-dy
      if (dabs(dxy).gt.0.5d0) j=j+idint(dsign(1d0,dxy)+0.5d0)
      dy=dfloat(j)+dy
```

```
      change=se*dy*t/x
      return
      end
      subroutine covel2(ilem,el,xv,par)
c
c     corrected variant
c     coordinates and velocity from elements
c      (duboshin 1, p.354 and 370)
c     input:  ilem-type of elements used((see subroutine inel)
c                  (if ilem.eq.3.or.lt.0, short-periodic perturbations
c                  are included)
c             el-array of elements as computed by function elemen.
c     output: xv-array of satellite rectangular coordinates(km)
c               and velocity(km/s)
c             par(1-3)-directional cosines
c             par(4-6)-first derivatives of cosines
c             par(7-9)-second derivatives of cosines
c
      implicit real*8 (a-h,o-z)
      dimension el(9),xv(6),par(9)
      data  rz, gm/6.37814d3,398601.3d0/
      if(ilem.eq.3) goto 3
      if(ilem.lt.0) goto 4
      dr=0d0
      di=0d0
      dl=0d0
      dm=0d0
      goto 10
    4 call short(dr,di,dl,dm,el)
      goto 10
    3 call elas (dr,di,dl,dm,el)
c     write(6,20) dr,di,dl,dm
c   20 format(1h ,80x,f8.3,3d11.3)
   10 sv=dsin(el(7))
      cv=dcos(el(7))
      au=el(7)+el(3)
      au=au+dl
      su=dsin(au)
      cu=dcos(au)
      el(1)=el(1)+dm
      sg=dsin(el(1))
      cg=dcos(el(1))
      el(2)=el(2)+di
      si=dsin(el(2))
      ci=dcos(el(2))
      pp=el(4)*(1d0-el(5)*el(5))
      cx=dsqrt(gm/pp)
      c2=1d0+el(5)*cv
      r=pp/c2
      r=r+dr
      par(1)=cu*cg-su*sg*ci
      par(2)=cu*sg+su*cg*ci
      par(3)=su*si
      par(4)=-su*cg-cu*sg*ci
      par(5)=-su*sg+cu*cg*ci
      par(6)=cu*si
      par(7)=sg*si
      par(8)=-cg*si
      par(9)=ci
      do 1 i=1,3
      xv(i)=r*par(i)
      xv(i+3)=cx*(par(i)*el(5)*sv+par(i+3)*c2)
    1 continue
      return
      end
      subroutine deltae(en,a,e,dn,da,de)
c
c     computation of missing parameters for subroutine inel (supposing
c     only stationary atmosphere's drag effect is involved)
c     input:  en-mean daily motion(rev/day)
c             a-semimajor axis(km)
c             e-eccentricity
c             dn-(1/2.dn) change of mean motion(rev/day2)
c     output: da-change of semimajor axis(km/day)
c             de-change of eccentricity(-/day)
c
      implicit real*8(a-h,o-z)
      real*8 en,a,e,dn,da,de
      de=0d0
      x=-4d0*dn/(3d0*en)
      da=x*a
      if(da.ge.0d0)return
      de=(1d0-e)*x
      return
      end
      subroutine detor(e,x)
c
c     corrected variant
c     elements from coordinates and velocity
c      (duboshin 1, p.292-296)
c     input: x-array of satellite coordinates and velocity(km,km/s).
c     subroutines: angle,antk
c     output: osculating elements:
c             e(1)-ascending node(rad)
c             e(2)-inclination(rad)
c             e(3)-argument of perigee(rad)
c             e(4)-semimajor axis(km)
c             e(5)-eccentricity(-)
c             e(6)-mean anomaly(rad)
c             e(7)-true anomaly(rad)
c             e(8)-mean daily motion(rad/day)
c             e(9)-(not computed)
c
      implicit real*8(a-h,o-z)
      dimension e(9),x(6),aj(21)
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
      data  rz,gm /6.37814d3,398601.3d0/
      data aj(2),aj(3),aj(4),aj(5)/1.082639d-3,-2.546d-6,-1.649d-6,
     *       -0.210d-6/
      r=dsqrt(x(1)*x(1)+x(2)*x(2)+x(3)*x(3))
      v=dsqrt(x(4)*x(4)+x(5)*x(5)+x(6)*x(6))
      cm=-gm/r
      c1=x(2)*x(6)-x(3)*x(5)
      c2=x(3)*x(4)-x(1)*x(6)
      c3=x(1)*x(5)-x(2)*x(4)
      c=dsqrt(c1*c1+c2*c2+c3*c3)
      f1=cm*x(1)+c3*x(5)-c2*x(6)
      f2=cm*x(2)+c1*x(6)-c3*x(4)
      f3=cm*x(3)+c2*x(4)-c1*x(5)
      f=dsqrt(f1*f1+f2*f2+f3*f3)
      e(1)=datan2(c1,-c2)
      e(2)=datan2(dsqrt(c1*c1+c2*c2),c3)
      p=c*c/gm
      e(5)=f/gm
      a=2d0/r-v*v/gm
      e(4)=1d0/a
      ci=dcos(e(2))
      sg=dsin(e(1))
```

```
      cg=dcos(e(1))
      if (dabs(ci).le.1d-15) goto 5
      sp=(-f1*sg+f2*cg)/ci
      cp= f1*cg+f2*sg
      su=(-x(1)*sg+x(2)*cg)/ci
      cu= x(1)*cg+x(2)*sg
      goto 6
    5 sp=f3
      su=x(3)
      if (dabs(cg).le.1d-2) goto 7
      cp=f1/cg
      cu=x(1)/cg
      goto 6
    7 cp=f2/sg
      cu=x(2)/sg
    6 u=datan2(su,cu)
      e(3)=datan2(sp,cp)
      e (7)=angle(u-e(3),dpi)
      e(6)=antk(e (7),e(5))
      aa=e(4)/rz
      aa=aa*aa
      dj2=1.5d0*aj(2)/aa
      an=dsqrt(gm/e(4)**3d0)*864d2
      e(8)=an*(1d0+dj2)
      return
      end
      subroutine elas(dr,di,dl,dm,el)
c     computation of shortperiodic perturbations caused by j2
c     (see osnovy teorii poleta kosmitcheskich apparatov,
c      moskva 1972, pp.347-8)
c     input:  el-array of elements as computed by function elemen.
c     output: dr-perturbation of the radius -vector(km)
c             di-perturbation of the inclination(rad)
c             dl-perturbation of the argument of latitude(rad)
c             dm-perturbation of the ascending node(rad)
c
      implicit real*8(a-h,o-z)
      real*8 dr,di,dl,dm
      dimension el(9)
      aj=66054.6d0
      au=el(7)+el(3)
      sp=dsin(el(3))
      cp=dcos(el(3))
      si=dsin(el(2))
      ci=dcos(el(2))
      s2i=dsin(2d0*el(2))
      aa=el(4)
      ae=el(5)
      su=dsin(au)
      cu=dcos(au)
      s2u=dsin(2.d0*au)
      c2u=dcos(2.d0*au)
      sv=dsin(el(7))
      cv=dcos(el(7))
      a=el(3)+2d0*el(7)
      sp2v=dsin(a)
      cp2v=dcos(a)
      a=2d0*el(3)+el(7)
      s2pv=dsin(a)
      c2pv=dcos(a)
      a=2d0*el(3)+3d0*el(7)
      s2p3v=dsin(a)
      c2p3v=dcos(a)
      p=aa*(1.d0-ae*ae)
      p=p*p
      a=c2u+ae*c2pv+ae*c2p3v/3d0
      a=a-(1d0+4d0*ae*cp/3d0)
      di=aj*s2i*a/p/4.d0
      a=-0.5*su+ae*(sv-0.5d0*s2pv-s2p3v/6d0)
      a=a+4d0*ae*sp/3d0
      dm=-aj*ci*a/p
      a=cu-ae*cv/2.d0-ae*c2pv/2d0-ae*cp2v/2d0
      a=a-(1d0-1.5d0*ae*cp)
      b=si*si
      c=a/(1.d0+ae*cv)+b*(c2u/6d0-2d0*cu/3d0+0.5d0)
      dr=aj*c/aa
      a=(-2.d0+4.d0*b/3.d0)*su+(-0.5d0+7.d0*b/12d0)*s2u+ae*(-2d0*b*sv+
     1(-1.5d0+5d0*b/6d0)*s2pv+(-1d0+b)*s2p3v/6d0+(-0.5d0+b/3d0)*sp2v)
      a=a+(5d0-14d0*b)*ae*sp/6d0
      dl=aj*a/p
      return
      end
      real function elemen*8(a,e,i)
c
c     computation of orbital element from time polynomials.
c     input:  a-array of the polynomial coefficients
c             e-time elapsed from epoch(days)
c             i-limiting power of e
c
      implicit real*8(a-h,o-z)
      dimension a(8)
      real*8 e
      b=a(1)
      k=2
    3 if(i-k)1,2,2
    2 b=b+a(k)*e**(k-1)
      k=k+1
      goto 3
    1 continue
      elemen=b
      return
      end
      subroutine elsol(tmj1,solf,alfas,dtasol)
c
c   calcul de l,angle solaire. precision la minute d,arc
c   ni abberation,ni nutation
c   solf=long soleil - alfas=alpha soleil - dtasol=delta soleil (de)
c
c     attention ce sous programme est en simple precision
c     seuls les arguments d appel et des commons sont en d p
c
      double precision tmj1,solf,alfas,dtasol
      rsde=4.848136e-06
      a1=((((99.+180.)*3600.)+41.*60.)+48.04)*rsde
      deupi=6.283185
      t=(tmj1-15019.5)/36525.
      ta=t*100.
      c1=aint(ta)
      c2=ta-c1
      b1=1.
      b2=2135902.e-11
      stoc=(b2*c2)+c2
      a3=((b2*c1)+stoc)*deupi
      a4=1.089*rsde*t*t
      gl=a1+a3+a4
      gl=amod(gl,deupi)
      if(gl.lt.0.0) gl=gl+deupi
      cdr=1.7453292e-02
```

```
      gld=gl/cdr
      a1=((281.*3600.)+(13.*60)+15.)*rsde
      a2=6189.03*rsde*t
      a4=0.012*(t**3)*rsde
      a3=1.63*rsde*t*t
      p=a1+a2+a3+a4
      p=amod(p,deupi)
      if(p.lt.0.0) p=p+deupi
      pd=p/cdr
      gm=gl-p
      gm=amod(gm,deupi)
      if(gm.lt.0.0) gm=gm+deupi
      gmd=gm/cdr
c
c  danjon  no 84  page 174
c
      v=gm+(6901.436*sin(gm)+72.158*sin(2.0*gm))*rsde
      glv=p+v
      glv=amod(glv,deupi)
      if(glv.lt.0.0) glv=glv+deupi
      soleil=glv
      z1=(.4308054e-01*sin(2*soleil))
      z2=(.93084227e-03*sin(soleil))
      z3=(.267097e-04*sin(6*soleil))
      z4=soleil-z1+z2-z3
      alfas = z4
      dtasol= asin(sin(soleil)*.397853)
      solf= soleil
      return
      end
      subroutine epler (em,ee,av,ae)
c
c     solution of the kepler equation
c     input:  em-mean anomaly (rad)
c             ee-eccentricity (-)
c     output: av-true anomaly(rad)
c             ae-eccentric anomaly(rad)
c
      implicit real*8 (a-h,o-z)
      real*8 em,ee,av,ae
      x=em
      ae=x
      if(dabs(dabs(x)-3.14159265) .le.1d-5) goto 1
    2 x1=x
      x=em+ee*dsin(x)
      if(dabs(x1-x) .gt. 1.e-10) goto 2
      ae=x
      a=(1d0+ee)/(1d0-ee)
      x=2d0*datan(dsqrt(a)*dtan(0.5d0*x))
    1 av=x
      return
      end
      subroutine inel(i)
c
c     input of elements in different codes (improved in 1980)
c     input:i-type of elements:
c             1-injection point parameters
c             2-nasa two-line elements
c             3-orbita code
c             4-five-line code
c             5-sao telex code
c             6-zipsat code
c             7-king-hele's code
c             8-orbit from position and velocity
c             9-tbus elements of noaa-n satellites
c          itype- input device specification(for standard input from car
c     common:elmn,type
c     subroutines:angle,antk,deltae,jayday,kozai,sapla,detor
c     output: ep-array of coefficients for arg.of perigee(rad)-tilljp1
c             eg-array of coefficients for asc. node(rad)-till jg1
c             ei-array of coefficients for inclination (rad)-till ji1
c             ee-array of coefficients for eccentricity(-)-till je1
c             ea-array of coefficients for semimajor axis(km)-till ja1
c             em-array of coefficients for mean anomaly(rev)-till jm1
c             djne-epoch of elements(mjd)
c             sate-name of the satellite
c     note 1: end of data set could be identified as jp1=0(generated
c                    by inel)
c     note 2: in sao format, variable ep(3) is used for storage of time
c             interval till next epoch of elements(this is used
c             in program sunrad2 etc.)
c
      implicit real*8(a-h,o-z)
      character*8 sate
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     1n1,jp1,jg1,ji1,je1,ja1,jm1
      common/type/ itype
      dimension aj(21),e(9),x(6)
      data  dhg,hg1/7.292115147d-5,6.300387487d0/
      data  rz,dmi/6.37814d3,398601.3d0/
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
      data aj(2),aj(3),aj(4),aj(5)/1.082639d-3,-2.546d-6,-1.649d-6,
     *        -0.210d-6/
      aj(1)=1.5*aj(2)
      it=itype
      if(itype.lt.8.or.itype.gt.100) it=55
      goto (1,2,3,4,5,6,7,8,9),i
c
c     injection point
c
    1 read(it,101,end=50) jne,a,b,c,sl,fi,ei(1),hp,ha,hn,j,sate
  101 format(i5,2f2.0,f5.3,3f7.2,3f8.1,i2,11x,a8)
      if(jne.eq.0) goto 50
      djne=(a*36d2+b*6d1+c)/864d2+jne
      ea(1)=(ha+hp)*0.5d0+rz
      ee(1)=(ha-hp)/(2d0*ea(1))
      a=ei(1)*cdr
      si=dsin(a)
      ci=dcos(a)
      sl=sl*cdr
      fi=fi*cdr
      if(dabs(si).ge.1d-5) goto   21
      ep(1)=dsin(sl)
      eg(1)=dsign(1d0,ci)*dsin(sl)
      go to 22
   21 ep(1)=dsin(fi)/si
      eg(1)=dtan(fi)*ci/si
      if(dabs(ep(1)).le.1d0) goto 22
c  wrong injection data-inclination is set equal to injection latitude
      a=dabs(fi)
      si=dsin(a)
      ci=dcos(a)
      ep(1)=dsign(1d0,ep(1))
      eg(1)=dsign(1d0,eg(1))
   22 ep(1)=dasin(ep(1))
      eg(1)=dasin(eg(1))
      em(1)=0d0
      c=ea(1)/rz
```

```
      b=1d0-aj(1)/(c*c)*(1d0-ee(1)*ee(1))**(-1.5d0)*(1d0-1.5d0*a*a)
      em(2)=dsqrt(b*dmi/ea(1)**3d0)*864d2/dpi
      if(j.eq.0) goto 23
      ep(1)=dpi*0.5d0-ep(1)
      eg(1)=dpi*0.5d0-eg(1)
   23 call sapla(djne,hg)
      eg(1)=sl-eg(1)+hg
      if(dabs(hn).le.1d0) go to 24
      b=rz+dabs(hn)
      c=ea(1)*(1d0-ee(1)*ee(1))/b-1d0
      if(ee(1).le.1d-6) go to 24
      b=c/ee(1)
      if(dabs(b).lt.1d0) goto 25
c  wrong injection data - injection point is supposed to be apogee
      em(1)=0.5d0
      ep(1)=ep(1)-em(1)*dpi
      go to 24
   25 em(1)=dsign(1d0,hn)*dacos(b)
      ep(1)=ep(1)-em(1)
      em(1)=antk(em(1),ee(1))/dpi
   24 ei(1)=a/cdr
      ep(1)=ep(1)/cdr
      eg(1)=eg(1)/cdr
      ep(2)=0d0
      eg(2)=0d0
      jp1=2
      jg1=2
      ji1=1
      ja1=1
      je1=1
      jm1=2
      n1=0
      goto 200
c
c nasa two-line elements
c
    2 read(it,102,end=50) sate,ny,day,em(3),em(4),nx,ei(1),eg(1),ee(1),
     *ep(1),em(1),em(2),n1
  102 format(9x,a8,1x,i2,f12.8,1x,f10.8,1x,f6.5,i2/
     18x,f8.4,f9.4,f8.7,2f9.4,f12.8,i5)
      if (ny.eq.0) goto 50
      call kozai(ei(1)*cdr,ee(1),em(2),ea(1),eg(2),ep(2))
      ny=ny+1900
      call jayday(0,1,ny,jne)
      djne=jne+day
      em(1)=em(1)/36d1
      em(3)=em(3)*0.5d0
      em(4)=em(4)*1d1**nx/6d0
      eg(2)=eg(2)*em(2)*36d1
      ep(2)=ep(2)*em(2)*36d1
      call deltae(em(2),ea(1),ee(1),em(3),ea(2),ee(2))
      jp1=2
      jg1=2
      ji1=1
      ja1=2
      je1=2
      jm1=4
      goto 200
c
c  orbita
c
    3 read(it,103,end=50) n1,jne,a,b,c,em(2),ea(1),ee(1),ei(1),eg(1),
     *ep(1),sate,em(3),ea(2),ee(2),ei(2),eg(2),ep(2)
  103 format(2i5,2f2.0,f5.3,f11.7,f9.3,f6.6,f7.4,2f8.4,7x,a5/5x,6f11.7)
      if(jne.eq.0) goto 50
      djne=(a*36d2+b*6d1+c)/864d2+jne-125d-3
      if(dabs(ep(2)).gt.1d-5.or.dabs(eg(2)).gt.1d-5) goto 13
   56 ci=dcos(ei(1)*cdr)
      aa=ea(1)*(1d0-ee(1)*ee(1))
      aa=6378.15d0/aa
      aa=1.62405d-3*aa*aa*36d1
      eg(2)=-aa*ci
      ep(2)=aa*(2.5d0*ci*ci-0.5d0)
   13 if(eg(1).lt.5d2) goto 14
      call sapla(djne,hg)
      eg(1)=eg(1)+hg/cdr-5d2
   14 em(2)=em(2)/144d1
      em(3)=em(3)/144d1
      a=em(2)-0.5d0*em(3)
      b=1/a
      c=-em(3)/(2d0*a**3d0)
      ep(3)=ep(2)*c
      ep(2)=ep(2)*b
      eg(3)=eg(2)*c
      eg(2)=eg(2)*b
      ei(2)=ei(2)*b
      ee(2)=ee(2)*b
      ea(2)=ea(2)*b
      em(4)=em(3)*em(3)/(2d0*a**5d0)
      em(3)=c
      em(2)=b
      em(1)=0d0
      if(dabs(ea(2)).lt.1d-5) call deltae(em(2),ea(1),ee(1),em(3),
     1ea(2),ee(2))
      jp1=3
      jg1=3
      ji1=2
      ja1=2
      je1=2
      jm1=4
      goto 200
c
c five-line
c
    4 read(it,104,end=50) sate,djne,em(1),eg(1),ep(1),ee(1),ei(1),em(2),
     *em(3),eg(2),ep(2),jdte,idte,ea(1)
  104 format(a8,g14.0,5g8.0/2g11.0,2g8.0,i6,i2,g10.0)
      if (djne.lt.1d1) goto 50
      ea(1)=ea(1)*rz
      em(1)=em(1)/36d1
      call deltae(em(2),ea(1),ee(1),em(3),ea(2),ee(2))
      jp1=2
      jg1=2
      ji1=1
      ja1=2
      je1=2
      jm1=3
      n1=0
      goto 200
c
c  sao telex code (changed in 1980 and 1985 = inclination format f15.10)
c
    5 read(it,*,end=50) sate,djne,ep(1),ep(2),eg(1),eg(2),ei(1),ei(2),
     *ee(1),ee(2),ep(3),em(1),em(2),em(3),em(4)
  105 format(a8,1x,f15.8,f15.10,f13.10,f15.10,f13.10/7x,f15.10,4f13.10/
     * 8x,f13.10,f14.10,1x,d11.4,1x,d11.4)
      if (djne.lt.1d1) goto 50
      call kozai(ei(1)*cdr,ee(1),em(2),ea(1),a,b)
```

```
      call  deltae(em(2),ea(1),ee(1),em(3),ea(2),a)
      jp1=2
      jg1=2
      ji1=2
      ja1=2
      je1=2
      jm1=4
      n1=0
      go to 200
c
c zipsat
c
    6 read(it,106,end=50) ny,day,ei(1),ep(1),em(1),em(2),em(3),nx,ee(1),
     *eg(1),sate
  106 format(6x,i2,f10.6,1x,3f7.7,f10.7,f6.5,i1,f8.7,f7.7,a8)
      if (ny.eq.0) goto 50
      call kozai(ei(1)*dpi,ee(1),em(2),ea(1),eg(2),ep(2))
      ny=ny+1900
      call jayday(0,1,ny,jne)
      djne=jne+day
      ep(1)=(ep(1)+0.25)*360d0
      ei(1)=ei(1)*360d0
      em(3)=em(3)*1d1**(-nx-2)
      eg(1)=eg(1)*360d0
      eg(2)=eg(2)*em(2)*360d0
      ep(2)=ep(2)*em(2)*360d0
      call deltae(em(2),ea(1),ee(1),em(3),ea(2),ee(2))
      jp1=2
      jg1=2
      ji1=1
      ja1=2
      je1=2
      jm1=3
      n1=0
      go to 200
c
c  king-hele
c
    7 read(it,107,end=50) n1,jne,ea(1),ee(1),ei(1),eg(1),ep(1),em(1),
     *em(2),em(3),em(4),sate
  107 format(2i5,f8.3,f5.5,f7.4,3f6.3,f8.4,2f5.5,9x,a5)
      if(jne.eq.0) goto 50
      djne=dfloat(jne)
      ep(2)=0d0
      eg(2)=0d0
      em(1)=em(1)/36d1
      em(2)=em(2)/36d1
      em(3)=em(3)/36d1
      em(4)=em(4)/36d1
      call deltae(em(2),ea(1),ee(1),em(3),ea(2),ee(2))
      jp1=2
      jg1=2
      ji1=1
      ja1=2
      je1=2
      jm1=4
      go to 200
c
c  orbit from position and velocity
c
    8 read(it,108,end=50) n1,jne,a,b,c,j,ax,ay,az,vx,vy,vz,sate,
     * em(2),em(3)
  108 format(2i5,2f2.0,f5.3,i1,6f9.3,1x,a5/5x,2f10.7)
      if(jne.eq.0) goto 50
      djne=(a*36d2+b*6d1+c)/864d2+jne-125d-3
      if (j) 51,51,52
   51 call sapla(djne,hg)
      shg=dsin(hg)
      chg=dcos(hg)
      x(1)=ax*chg-ay*shg
      x(2)=ax*shg+ay*chg
      x(3)=az
      x(4)=vx*chg-vy*shg-dhg*x(2)
      x(5)=vx*shg+vy*chg+dhg*x(1)
      x(6)=vz
      goto 53
   52 x(1)=ax
      x(2)=ay
      x(3)=az
      x(4)=vx
      x(5)=vy
      x(6)=vz
   53 call detor(e,x)
      ep(1)=e(3)/cdr
      ep(2)=0d0
      eg(1)=e(1)/cdr
      eg(2)=0d0
      ei(1)=e(2)/cdr
      ee(1)=e(5)
      ea(1)=e(4)
      if (em(2).le.1d-1) goto 55
      ei(2)=0d0
      i=3
      goto 56
   55 em(1)=e(6)/dpi
      em(2)=e(8)/dpi
      jp1=2
      jg1=2
      ji1=1
      je1=1
      ja1=1
      jm1=2
      goto 200
c
c tbus elements - noaa satellites
    9 read(it,*,end=50) y,rm,day,h,rmm, s,ee(1),ep(1),eg(1),ei(1),
     &ea(1),em(1)
      if(y.eq.0d0) go to 9
      ea(1)=ea(1)/(1d0-0.5*aj(2)*(rz/ea(1))**2*
     *(1d0-1.5d0*dsin(ei(1)*cdr)**2))
      em(2)=sqrt(dmi/ea(1)**3)*86400d0/dpi
      call kozai(ei(1)*cdr,ee(1),em(2),ea(1),eg(2),ep(2))
      ny=y
      ny=ny+1900
      month=rm
      iden=day
      call jayday(iden,month,ny,jne)
      djne=jne+(h+rmm/6d1+s/36d2)/24d0
      em(1)=em(1)/36d1
      eg(2)=eg(2)*em(2)*36d1
      ep(2)=ep(2)*em(2)*36d1
      jp1=2
      jg1=2
      ji1=1
      ja1=1
      je1=1
      jm1=2
      goto 200
```

```
c
  200 if(dabs(ep(2)).gt.1d-5.or .dabs(eg(2)).gt.1d-5) goto 204
      call kozai(ei(1)*cdr,ee(1),em(2),a,b,c)
      if(ea(1).lt.1d0)ea(1)=a
      ep(2)=c*em(2)*36d1
      eg(2)=b*em(2)*36d1
  204 eg(1)=angle(eg(1),360d0)
      ep(1)=angle(ep(1),360d0)
      do 201 k=1,jp1
  201 ep(k)=ep(k)*cdr
      do 202 k=1,jg1
  202 eg(k)=eg(k)*cdr
      do 203 k=1,ji1
  203 ei(k)=ei(k)*cdr
      return
   50 jp1=0
      return
      end
      subroutine jayday(nd,nm,ny,jnl)
c
c     gives jnl=days since nov.1g.0 ut 1858 corres-
c     ponding to gregorian calendar month,day and year.
c
      if(ny.lt.1901)go to 10
      if(ny.lt.2100)go to 11
   10 jnl=0
      go to 14
   11 mm=nm
      my=ny-1900
      if(mm.gt.2)go to 12
      mm=mm+9
      my=my-1
      goto 13
   12 mm=mm-3
   13 jnl=15078+(1461*my)/4+(153*mm+2)/5+nd
   14 return
      end
      subroutine kozai(angr,ecc,aprm1,cp,grr,grp)
c
c     computation of missing parameters for inel (sao spec.rep.no.30)
c     input:  angr-inclination(rad)
c             ecc-eccentricity(-)
c             aprm1-mean daily motion(rev/day)
c     output: cp-semimajor axis(km)
c             grr-rate of change of ascension node(rev/rev)
c             grp-rate of change of argument of perigee(rev/rev)
c
      implicit real*8(a-h,o-z)
      a=aprm1*aprm1
      f=(0.7537138d+14/a)**(1d0/3d0)
      x=ecc*ecc
      y=1d0-x
      aa=dsqrt(y)
      ab=f*y*f*y
      ac=0.6606372d+5/ab
      sngr=dsin(angr)
      cngr=dcos(angr)
      ae=sngr*sngr
      w=0.7537138d+14*(1d0-ac*aa*(1d0-1.5d0*ae))
      r=w/a
   10 t=f*f
      u=3d0*t
      v=f*t
      dr=r-v
      f=f+dr/u
      if(dabs(dr).gt.0.5d-06*r)go to 10
      cp=f
      ap=ae*ae
      ah=cngr*cngr
      ai=ah*ah
      aj=0.119392d+11/(ab*ab)
      grr=-(ac*cngr*(1d0+ac*(1.5+x/6d0-2d0*aa-ae*(5d0/3d0-5d0*x/24d0
     1-3d0*aa)))+aj*cngr*(6d0/7d0-3d0*ae/2d0)*(1d0+3d0*x/2d0))
      grp=ac*(2d0-2.5*ae)*(1d0+ac*(2d0+x/2d0-2d0*aa-ae*(43d0/24d0
     1-x/48d0-3d0*aa)))-5d0*ac*ac*x*ai/12d0+aj*(12d0/7d0-93d0*ae/14d0
     2+21d0*ap/4d0+x*(27d0/14d0-189d0*ae/28d0+81d0*ap/16d0))
      return
      end
      subroutine krizak(a,nn1,nn2,n,np,j)
c
c reseni linearnich rovnic o n neznamych a p pravych stranach metodou
c krizoveho nasobeni (np=n+p). rozsirena matice soustavy je umistena
c v poli a, pri vystupu jsou neznane ulozeny po sloupcich pocinaje n+1
c sloupcem. jeli j ruzne od nuly je prvnich j sloupcu linearne zavislych
c matice a se vypoctem znici.
c
      implicit real*8(a-h,o-z)
      dimension a(nn1,nn2)
      n1=n-1
      do 3 j=1,n
      max=1
      am=dabs(a(1,j))
      nj=n-j+1
      do 1 i=1,nj
      if(dabs(a(i,j)) .le. am) go to 1
      max=i
      am=dabs(a(i,j))
    1 continue
      if (am .eq. 0d0) return
      am=a(max,j)
      a(max,j)=a(1,j)
      j1=1+j
      do 3 k=j1,np
      a1=a(max,k)
      a(max,k)=a(1,k)
      do 2 i=1,n1
    2 a(i,k)=(am*a(i+1,k)-a(i+1,j)*a1)/am
    3 a(n,k)=a1/am
      j=0
      return
      end
      subroutine longps(nn,ai,eo,an,b,otm,otv,p)
c
c     computation of longperiodic and secular perturbations from j2-j5
c     with basic period (see sel/i,p.130).
c     input:  nn-limiting value of jn   (max.5)
c             ai-inclination (rad)
c             eo-eccentricity (-)
c             an-mean daily motion (rev/d)
c             b -semimajor axis (km)
c     output: otm-secular change of perigee(rad/day)
c             otv-secular change of ascending node(rad/day)
c             p(1)-amplitude of longperiodic changes of arg.perigee(rad)
c             p(2)-amplitude of ascending node (rad)
c             p(3)-amplitude of inclination(rad)
c             p(4)-amplitude of eccentricity(-)
c             p(5)-amplitude of mean anomaly at epoch(rad)
c
```

```
      implicit real*8 (a-h,o-z)
      dimension c(10),e(10),h(10),g(10),p(15),q(10),a(10),aj(21)
      data aj(2),aj(3),aj(4),aj(5)/1.082639d-3,-2.546d-6,-1.649d-6,
     *       -0.210d-6/
      data  rz,dmi/6.37814d3,398601.3d0/
      n=nn
      aa=b/rz
      su3=float(n)/2
      n=n/2-1
      co=dcos(ai)
      c(1)=co*co
      e(1)=eo*eo
      if(n)115,115,116
  116 do 110 j=1,n
      c(j+1)=c(j)*c(1)
      e(j+1)=e(j)*e(1)
  110 continue
  115 ee=1.-e(1)
      et=dsqrt(ee)
      si=dsin(ai)
      s2=si*si
      c5=1.-5.*c(1)
      c9=8.+c5
      g2=3./(-128.)*(2.*c5*(5.+43.*c(1))+(25.-126.*c(1)+145.*c(2))*e(1)
     1-24.*c5*(1.-3.*c(1))*et)
      h2=3./(-32.)*(4.*(1.-10*c(1))-c9*e(1)+12.*(1.-3.*c(1))*et)
      n=n+1
      go to (102,103),n
  103 h(2)=15./32.*(3.-7.*c(1))*(2.+3.*e(1))
      g(2)=15./128.*(12.-144.*c(1)+196.*c(2)+e(1)*(9.-126.*c(1)+189.*
     1c(2)))
  102 h(1)=1.5
      g(1)=-0.75*c5
      po=aa*ee
      su1=0.
      su2=0.
      p2=po*po
      pj=1.
      do 101 j=1,n
      pj=pj*p2
      ajp=aj(2*j)/pj
      su1=su1+ajp*g(j)
      su2=su2+ajp*h(j)
  101 continue
      pj=p2*p2
      ajp=aj(2)*aj(2)/pj
      otm=an*(su1+ajp*g2)*6.28318531
      otv=(-an*co)*(su2+ajp*h2)*6.28318531
      su3=su3-n
      su3=dabs(su3)-0.1
      if(su3)120,120,121
  120 n=n-1
  121 go to (113,114),n
  114 g(2)=1.-14.*c(1)+21.*c(2)
      h(2)=28.-84.*c(1)
      pj=5./32.
      p(2)=(4.+3.*e(1))*pj
      q(2)=pj*(4.+25.*e(1)+6.*e(2))
      a(2)=pj*g(2)*(4.+9.*e(1))
  113 g(1)=c5
      h(1)=10.*c5*c5
      p(1)=0.5
      q(1)=0.5
      a(1)=0.5*c5
      su1=0.
      su2=0.
      su3=0.
      su4=0.
      pj=1./po
      do 111 j=1,n
      jj=2*j+1
      pj=pj*p2
      ajp=aj(jj)/pj
      ajq=ajp*q(j)
      ajp=ajp*p(j)
      su1=su1+ajp*g(j)
      su2=su2+ajp*(c9*g(j)-s2*c5*h(j))
      su3=su3+ajq*g(j)
      su4=su4+ajp*a(j)
  111 continue
      pj=aj(2)*c5
      if(dabs(pj)-0.1e-9)127,127,128
  127 pj=1.0e15*dsign(1.d0,si)/dsign(1.d0,pj)
      go to 129
  128 pj=si/pj
  129 p(4)=pj*(-su1)*ee
      p(1)=pj*su3/eo
      p(5)=pj*su4*et*et*et/(eo*6.28318531)
      if(dabs(si)-0.1e-6)122,122,123
  122 pj=1.0e15*dsign(1.d0,co)
      go to 124
  123 pj=eo*co/si
  124 p(3)=pj*(-p(4))/et
      if(dabs(c5)-0.1e-6)125,125,126
  125 p(2)=1.0e18*dsign(1.d0,pj)*dsign(1.d0,su2)/dsign(1.d0,aj(2))
      goto 130
  126 p(2)=pj*su2/(aj(2)*c5*c5)
  130 p(1)=co*(-p(2))-p(1)
      do 100 i=6,15
  100 p(i)=0d0
      return
      end
      subroutine nhms(nh,nm,ns,c,i)
c
c     transformation from fractions of day into hours,minutes
c     and seconds.
c     input : c-fractions of day
c             i-if equal to 1,result is rounded to integral seconds,
c               -if equal to -1,result is rounded to integral minutes.
c     output : nh-hours
c              nm-minutes
c              ns-seconds(rounded)
c
      implicit real*8(a-h,o-z)
      ns=0
        t=dabs(c*24d0)
      nh=idint(t)
      t=dmod(t,1d0)*6d1
      if(i)1,1,2
    1 nm=idint(t+0.5d0)
      go to 4
    2 nm=idint(t)
      s=dmod(t,1d0)*6d1
      ns=idint(s+0.5d0)
      if(ns.lt.60) go to 3
      nm=nm+1
      ns=0
    4 if(nm.lt.60) go to 3
```

```
      nh=nh+1
      nm=0
    3 if(c.lt.0d0) nh=-nh
      return
      end
      subroutine orbnas (ilem)
c
c     transformation of input elements from code orbita (ilim=3)
c     into nasa two-line code (ilem=2)
c     if ilem.lt.o punch the resulting elements
c     attention - value of ilem is changed during execution
      implicit real*8(a-h,o-z)
      character*8 sate
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,
     &sate,n1,jp,jg,ji,je,ja,jm
      data c20,rz/-1082.6d-6,6378.14d0/
      dpi=datan2(0d0,-1d0)*2d0
      j23r=-7.173247d0
      rod=36d1/dpi
      em(1)=antk(-ep(1),ee(1))*rod
      alfa=1.5d0*(rz/ea(1))**2*((2.5d0*dcos(ei(1))**2-0.5d0)/
     &((1d0+ee(1)*dcos(ep(1)))**2*dsqrt(1d0-ee(1)**2))
     &+(1d0+ee(1)*dcos(ep(1)))**3/(1d0-ee(1)**2)**3)
      t0=1440d0/em(2)/(1d0+c20*alfa)
      tome=t0*(1d0+1.5d0*c20*(rz/ea(1))**2*((1d0+ee(1)*
     &dcos(-ep(1)))/(1d0-ee(1)**2))**3)
      em(2)=1440d0/tome
      ee(1)=ee(1)+j23r/ea(1)*dsin(ei(1))*dsin(ep(1))
      ei(1)=ei(1)*rod
      eg(1)=eg(1)*rod
      ep(1)=ep(1)*rod
      jnem=idint(djne)
      call calday(nd,nm,nrok,jnem)
      call jayday( 0, 1,nrok,jnem1)
      porden=djne-jnem1
      ny=nrok-1900
      write(66,200) sate,ny,porden,em(3),ei(1),eg(1),ee(1),ep(1),
     &em(1),em(2)
  200 format(' ',' elements in nasa two-line code',
     &//9x,a8,1x,i2,f12.8,1x,f10.8,/8x,f8.4,f9.4,f8.7,
     &2f9.4,f12.8)
      if (ilem.lt.0)
     *write(7,201) sate,ny,porden,em(3),ei(1),eg(1),ee(1),ep(1),
     &em(1),em(2)
  201 format(9x,a8,1x,i2,f12.8,1x,f10.8,1x/8x,f8.4,f9.4,f8.7,
     &2f9.4,f12.8)
      ei(1)=ei(1)/rod
      eg(1)=eg(1)/rod
      ep(1)=ep(1)/rod
      em(1)=em(1)/36d1
      ilem=2
      call outel(ilem)
      return
      end
      subroutine outel(i)
c
c     print of elements obtained by inel.
c     angular elements are printed in (degrees) or (degrees/day),
c     ea in (km), em in (rev) or (rev/day).
c      input: i-type of elements
c     common:elmn
c      subroutines: ahms,calday
c      output: (none)
c
      implicit real*8(a-h,o-z)
      character*8 sate
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     1n1,jp1,jg1,ji1,je1,ja1,jm1
      character fff
      dimension ap(3),ag(3),ai(2)
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
c
      fff=char(12)
      write(66,100) sate
  100 format('   satellite : ',a8/'   source of elements : ')
      goto (1,2,3,4,5,6,7,8,9,10),i
    1 write(66,11)
   11 format(' ',27x,'injection point'/)
      goto 200
    2 write(66,12)
   12 format(' ',27x,'nasa two-line elements'/)
      go to 200
    3 write(66,13)
   13 format(' ',27x,'orbita'/)
      em(1)=0d0
      go to 200
    4 write(66,14)
   14 format(' ',27x,'five-line code'/)
      go to 200
    5 write(66,15)
   15 format(' ',27x,'sao telex code'/)
      go to 200
    6 write(66,16)
   16 format(' ',27x,'zipsat'/)
      go to 200
    7 write(66,17)
   17 format(' ',27x,'king-hele'/)
      goto 200
    8 write(66,18)
   18 format(' ',27x,'position and velocity'/)
      goto 200
    9 continue
      goto 200
   10 write(66,20)
   20 format(' ',27x,'program prior'/)
      goto 200
  200 jne=idint(djne)
      ajne=dmod(djne,1d0)
      call calday(md,mm,my,jne)
      call ahms(nh,nm,s,ajne)
      do 201 k=1,jp1
  201 ap(k)=ep(k)/cdr
      do 202 k=1,jg1
  202 ag(k)=eg(k)/cdr
      do 203 k=1,ji1
  203 ai(k)=ei(k)/cdr
c
      if(n1.gt.0) write(66,204)n1
  204 format(' ',2x,'orbit : ',i11)
      write(66,205)jne,ajne,md,mm,my,nh,nm,s
  205 format(' ',2x,'epoch : ',6x,i5,3x,f8.6,2x,2(i2,'.'),i4,
     12x,2(i2,':'),f5.2)
      write(66,206)(ap(k),k=1,jp1)
  206 format(' ' ,2x,'perig : ',2f11.6,2d14.6)
      write(66,207)(ag(k),k=1,jg1)
  207 format(' ',2x,'rnode : ',2f11.6,2d14.6)
      write(66,208)(ai(k),k=1,ji1)
  208 format(' ',2x,'incln : ',2f11.6,2d14.6)
```

```
      write(66,209)(ee(k),k=1,je1)
  209 format(' ',2x,'eccen : ',2f11.6,2d14.6)
      write(66,210)(ea(k),k=1,ja1)
  210 format(' ',2x,'saxis : ',2f11.3,2d14.6)
      write(66,211)(em(k),k=1,jm1)
  211 format(' ',2x,'manom : ',2f11.6,2d14.6)
      if(i.eq.3) write(66,212)
  212 format(' ',12x,'(mean nodal motion is given)')
      write(66,213)
  213 format(' ')
      return
      end
      subroutine phase(xv,xst,shg,chg,el,ra,stb,lgt,a)
c
c     computation of the illumination conditions
c     input:  xv-array of geocentric satellite coordinates(km)
c               and velocity(km/s)
c             xst-array of station coordinates(km)
c             shg-sinus of sidereal time hg
c             chg- cosinus of sidereal time hg
c             el-longitude of the sun (rad)
c             ra-distance satellite station(km)
c             stb-magnitude of the satellite at culmination
c     subroutines: vect
c     output :  lgt.eq.s satellite is in shadow
c               lgt.eq.8 brightness is.le.1 mg. than maximum
c               lgt.eq.6 brightness is.le.2 mg. than maximum
c               lgt.eq.4 brightness is.le.3 mg. than maximum
c               lgt.eq.2 brightness is.le.4 mg. than maximum
c               lgt.eq.1 brightness is.lt.4 mg. than maximum
c             a-not rounded value of lgt(if lgt.eq.s,a=-100)
c
      implicit real*8(a-h,o-z)
      dimension xv(6),xst(12),i(10)
      data se,ce/0.397985d0,0.917392d0/,i/' 1  ',' 2  ',' 3  ',' 4  ',
     1' 5  ',' 6  ',' 7  ',' 8  ',' 9  ',' s* '/
      xs=dcos(el)
      a=dsin(el)
      ys=a*ce
      zs=a*se
      xp=-xv(1)
      yp=-xv(2)
      zp=-xv(3)
      r=vect(xp,yp,zp)
      ca=(xp*xs+yp*ys+zp*zs)/r
      a=dacos(ca)
      if(ca.le.0d0) go to 1
      if (r*dsin(a).gt.6370d0) go to 1
      a=-1d2
      lgt=i(10)
      return
    1 xp=xp+xst(1)*chg-xst(2)*shg
      yp=yp+xst(1)*shg+xst(2)*chg
      zp=zp+xst(3)
      r=vect(xp,yp,zp)
      ca=(xp*xs+yp*ys+zp*zs)/r
      f=0.5d0*(ca+1d0)
      d=5d0*dlog10(ra)-2.5d0*dlog10(f)
      a=2d0*(stb-d)
      n=max0(1,idint(a))
      lgt=i(n)
      return
      end
      subroutine posic2(enl,i,k,n,x,u,n3)
c
c     computation of geocentric coordinates for given elapsed time
c      input:enl-elapsed  time from epoch(days)
c            i-type of elements(1-7).
c              (if.le.0.or.eq.3,shortperiodic pert. are included)
c     for pc are not included!
c            k-if.gt.0, tesseral pert. are included
c            n-index for perturbation array dal
c      common: elmn,el
c      subroutines:angle,antk,covel2,elemen,epler
c     output for covel2:array of actual values of elements:
c             el(1)-ascending node(rad)
c             el(2)-inclination(rad)
c             el(3)-argument of perigee(rad)
c             el(4)-semimajor axis(km)
c             el(5)-eccentricity(-)
c             el(6)-mean anomaly(rad)
c             el(7)-true anomaly(rad)
c             el(8)-(not computed)
c             el(9)-eccentric anomaly(rad)
c      output:x-array of satellite coordinates and velocity.
c             u-argument of latitude(rad)
c             n3-orbit number(if used in orbital elements).
c
      implicit real*8(a-h,o-z)
      character*8 sate
      dimension el(9),x(6),par(9)
      common/el/ al(100,8),dal(100,6)
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     1n1,jp1,jg1,ji1,je1,ja1,jm1
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
      el(1)=elemen(eg,enl,jg1)
      el(2)=elemen(ei,enl,ji1)
      el(3)=elemen(ep,enl,jp1)
      el(4)=elemen(ea,enl,ja1)
      el(5)=elemen(ee,enl,je1)
      if(el(5).lt.0d0) el(5)=0d0
      n3=0
      a=0d0
      am=elemen(em,enl,jm1)
      if(n1.le.0) goto 1
      n2=idint(am)
      if(am.lt.0d0) n2=n2-1
      n3=n1+n2
    1 continue
      am=dmod(am,1d0)*dpi
      if(i.eq.3) a=antk( el(3),el(5))
      el(6)=angle(am-a,dpi)
      if(k.eq.0)goto 2
      el(1)=el(1)+dal(n,2)
      el(2)=el(2)+dal(n,3)
      el(3)=el(3)+dal(n,1)
      el(4)=el(4)+dal(n,6)*1d3
      el(5)=el(5)+dal(n,4)
      el(6)=el(6)+dal(n,5)
    2 call epler(el(6),el(5),el(7),el(9))
      u=angle(el(3)+el(7),dpi)
      call covel2(i,el,x,par)
      return
      end
      subroutine sapla(dmjd,sod)
c
c     evaluation of the mean sidereal time according to newcomb
c     input: dmjd-modified julian date
```

```
c     output: sod-sidereal time at the greenwich meridian(rad)
c
      implicit real*8 (a-h,o-z)
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
      data  dhg,hg1/7.292115147d-5,6.300387487d0/
      mjd=idint(dmjd)
      sod=0.2769194d0+0.00273790926d0*( mjd-15019.5d0)
      sod=dmod(sod,1d0)*dpi+hg1*dmod(dmjd,1d0)
      return
      end
      subroutine sbg1(n,cas,a,h,rych,ap,zp,delta,rm,sigma,kod)
c
c vypocet nastaveni 4 ose mantaze,rychlosti pohybu kolem 3. osy
c vstup :n    ... (i4)pocet zadanych bodu
c        cas ... (pole r8)pole casu - mjd vcetne zlomku
c        a   ... (pole r8)pole azimutu
c        h   ... (pole r8)pole vysek
c        rych .. (pole r8)rychlost (radiany za sec.)
c vystup:ap,zp,delta ... (r8)nastaveni 1.,2. a 4. osy
c        rm  ... (r8)stredni chyba urceni delta
c        sigma . (pole r8)pole nastaveni 3. osy
c        kod ... (i4)kod= 0 pomalu doprava  kod=10 rychle doprava
c                   kod= 1 pomalu doleva   kod=11 rychle doleva
c nutna procedura:krizak
c
      implicit real*8(a-h,o-z)
      dimension p(3,4),cas(1),a(1),h(1),sigma(1),rych(1),del(90)
      data pi/3.1415926535897324d0/
      ap=0d0
      zp=0d0
      delta=0d0
      rm=0d0
      kod=0
      do 1 i=1,n
      if(dabs(rych(i)).gt.0.024d0) kod=10
    1 sigma(i)=0d0
      if(n.lt.4) return
      fa=dcos(h(1))*dsin(a(1))
      fb=dcos(h(1))*dcos(a(1))
      fc=dsin(h(1))
      ns=n/2+1
      al21=dcos(h(ns))*dsin(a(ns))-fa
      be21=dcos(h(ns))*dcos(a(ns))-fb
      ga21=dsin(h(ns))-fc
      al31=dcos(h(n))*dsin(a(n))-fa
      be31=dcos(h(n))*dcos(a(n))-fb
      ga31=dsin(h(n))-fc
      x1=al21*ga31-al31*ga21
      x2=be31*ga21-be21*ga31
      ap=datan2(x2,x1)
      ctgzp=-al21*dsin(ap)/ga21-be21*dcos(ap)/ga21
      if(ctgzp.ge.0d0) go to 4
      ap=dmod(ap+pi,2d0*pi)
      ctgzp=-ctgzp
    4 zp=datan2(1d0,ctgzp)
      sz=dsin(zp)
      cz=dcos(zp)
      s1=0d0
      do 3 i=1,n
      del(i)=dasin(cz*dsin(h(i))+sz*dcos(h(i))*dcos(a(i)-ap))
    3 s1=s1+del(i)
      delta=s1/dfloat(n)
 c reseni normalnich rovnic
      do 11 i=1,3
      do 11 j=1,4
   11 p(i,j)=0d0
      p(1,1)=dfloat(n)
      do 12 i=1,n
      b=(sz*dsin(h(i))-cz*dcos(h(i))*dcos(a(i)-ap))/dcos(del(i))
      c=-sz*dcos(h(i))*dsin(a(i)-ap)/dcos(del(i))
      d=delta-del(i)
      p(1,2)=p(1,2)+b
      p(1,3)=p(1,3)+c
      p(1,4)=p(1,4)+d
      p(2,2)=p(2,2)+b*b
      p(2,3)=p(2,3)+b*c
      p(2,4)=p(2,4)+b*d
      p(3,3)=p(3,3)+c*c
   12 p(3,4)=p(3,4)+c*d
      p(2,1)=p(1,2)
      p(3,1)=p(1,3)
      p(3,2)=p(2,3)
      call krizak(p,3,4,3,4,j)
      if(j.ne.0) write(66,601)j
  601 format(' prvnich',i2,' sloupcu linearne zavislych (podprogram sbg)
     1')
      delta=delta+p(1,4)
      zp=zp+p(2,4)
      ap=ap+p(3,4)
      sz=dsin(zp)
      cz=dcos(zp)
      s1=0d0
      do 13 i=1,n
      sigma(i)=dasin(-dsin(a(i)-ap)*dcos(h(i))/dcos(del(i)))
      del(i)=dasin(cz*dsin(h(i))+sz*dcos(h(i))*dcos(a(i)-ap))
      v=delta-del(i)
   13 s1=s1+v*v
      rm=dsqrt(s1/(dfloat(n)*dfloat(n-1)))
      if(sigma(2).lt.sigma(1)) kod=kod+1
      return
      end
      subroutine short(dr,di,dl,dm,el)
c
c     computation of the short periodic perturbations by kozai's
c     formula (see sel/i,p.146)
c     input:el-array of elements,computed by function elemen.
c     subroutines : angle
c     output: dr-perturbation of the radius -vect(km)                0
c             di-perturbation of the inclination(rad)
c             dl-perturbation of the argument of latitude(rad)
c             dm-perturbation of the ascending node(rad)
c
      implicit real*8 (a-h,o-z)
      dimension el(9),aj(21)
      data aj(2),aj(3),aj(4),aj(5)/1.082639d-3,-2.546d-6,-1.649d-6,
     *       -0.210d-6/
      data  rz,dmi/6.37814d3,398601.3d0/
      data twopi,cdr/6.283185307179585d0,1.745329252d-2/
      eo=el(5)
      om=el(3)
      aa=el(4)/rz
      ce=dcos(el(9))
      am=el(6)
      v=el(7)
      sv=dsin(v)
      cv=dcos(v)
      ee=1.-eo*eo
      et=dsqrt(ee)
```

```
      si=dsin(el(2))
      co=dcos(el(2))
      s2=si*si
      po=aa*ee
      h2=1.-eo*ce
      g2=1.5*s2-1.
      pj=1.5d0*aj(2)/po
      p2=eo/(1.+et)
      u=2.*om+v
      u2=u+v
      u3=u2+v
      cu2=dcos(u2)
      dr=pj/3.*(g2*(1.-h2/et+p2*cv)+0.5*cu2*s2)
      dr=dr*rz
      pj=pj/po
      di=0.5*si*pj*co*(cu2+eo*(dcos(u)+1./3.*dcos(u3)))
      cu2=dsin(u2)
      su=dsin(u)
      su3=dsin(u3)
      h2=angle(v-am+eo*sv,-twopi)
      dl=pj*(0.5*((7./6.*s2-1.)*cu2+eo*((5./3.*s2-1.)*su+(s2-1.)
     1*su3/3.))-g2/3.*((1.-et)*sv*cv+eo*p2*p2*sv)+(2.5*s2-2.)*h2)
      dm=pj*co*(0.5*(cu2+eo*(su+su3/3.))-h2)
      return
      end
      real function vect*8 (x,y,z)
      real*8 x,y,z
c
c     lenght of the vector with
c     components x,y,z
c
      vect=dsqrt(x*x+y*y+z*z)
      return
      end
      double precision function dthg83(dmj)
c
c  pocita pravy greenw. hvezdny cas pro okamzik  dmj
c  entry dthgm - pocita stredni hvezdny cas
c  nutna procedura nuta83 (pocita nutaci)
c
      implicit real*8(d)
      common/nu/dmi,d(4)
      call nuta83(dmj)
      dthg83=dmi
      goto 1
      entry dthgm(dmj)
      dthg83=0.
    1 dvepi=6.2831853071796d0
      dzl=dmod(dmj,1d0)
      dt=(dmj-dzl-51544.5)/36525.
      t=dt
      dzl=(dble((-5.9e-15*t+5.9006e-11)*t)+1.00273790935d0)*dzl*dvepi
      dth=(-6.2e-6*t+9.3104e-2)*t
      dth=dmod(((dth+8640184.812866d0)*dt+24110.54841d0)/
     /13750.987083134d0+dthg83+dzl,dvepi)
      if(dth.lt.0d0) dth=dvepi+dth
      dthg83=dth
      return
      end
      subroutine nuta83(dmj)
c
c  pocita nutaci podle wahra pro modifikovany juliansky okamzik  dmj
c  vystup: dmi=dpsi*cos(deps), dni=dpsi*sin(deps)
c          ddeps, dpsi - nutace ve sklonu a delce
c          deps - okamzity sklon ekliptiky
c
      implicit real*8(d)
      real fdl(5)
      integer arg(106),p(106),e(106),po(15),pt(15),eo(15),et(15),no(15)
      common/nu/dmi,dni,ddeps,dpsi,deps
c  argumenty
      data arg/55556,55557,35756,75355,35757,64545,53736,75356
     1,55737,56555,56737,54737,55736,75535,55735,57555,56556
     2,57737,54556,35576,54736,75536,56736,65545,76535,55376
     3,56375,56557,45566,56735,55757,65555,55756,65757,65535
     4,45757,55575,65556,45556,45777,65756,55777,75555,65737
     5,75757,55755,45756,45576,65536,45776,66535,56757,54757
     6,65777,65575,75737,55576,55776,65736,55536,64555,75756
     7,56535,65355,55565,66555,65755,64757,44777,35556,85757
     8,54777,66757,45736,75556,65557,85555,55767,45557,65515
     9,35777,45797,75515,66737,65776,35797,45957,64535,75736
     a,75777,65576,55937,85737,65735,56756,44576,55356,55747
     b,56575,65335,54756,66536,65375,75575,55797,56565/
c  koeficienty nutace v delce (0.0001")
      data p/0,0,46,11,-3,-3,-2,1,5*0,48,-22,0,-15,0,-12,-6,-5
     1,4,4,-4,1,1,-1,1,1,-1,4*0,-158,123,63,2*0,-59,-51,-38,29
     2,29,-31,26,21,16,-13,-10,-7,7,-7,-8,6,6,-6,-7,6,-5,5,-5,-4
     3,4,-4,-3,3,-3,-3,-2,-3,-3,2,-2,2,-2,2,2,1,-1,1,-2,-1,1,-1,-1
     4,1,1,1,-1,-1,1,1,-1,1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,1/
c  koeficienty nutace ve sklonu (0.0001")
      data e/0,0,-24,0,1,0,1,0,5*0,1,0,0,9,0,6,3,3,-2,-2,7*0,4*0
     1,-1,-53,-2,2*0,26,27,16,-1,-12,13,-1,-10,-8,7,5,0,-3,3,3,0
     2,-3,3,3,-3,3,0,3,5*0,5*1,-1,1,-1,1,0,-1,-1,0,-1,1,0,-1,1,1
     3,0,0,-1,17*0/
c  indexy a koeficienty casove zavislych clenu (0.0001")
      data no/1,2,9,10,11,12,13,16,18,31,32,33,34,38,39/
      data po/-171996,2062,-13187,1426,-517,217,129,17,-16,-2274
     +,712,-386,-301,63,-58/
      data eo/92025,-895,5736,54,224,-95,-70,0,7,977,-7,200,129,-33,32/
c  casove zmeny (0.00001")
      data pt/-1742,2,-16,-34,12,-5,1,-1,1,-2,1,-4,0,1,-1/
      data et /89,5,-31,-1,-6,3,0,0,0,-5,0,0,-1,0,0/
c
      dvepi=6.283185308d0
      drovt=206264.80625d0
      dt=(dmj-51544.5)/36525.
      t=dt
c  fdl(i) = omega, d, f, l1, l  (v otockach)
      fdl(1)=dmod(((6.2d-9*dt+5.752d-6)*dt-5.3726007245d0)*dt+
     +0.3473458951d0,1d0)*dvepi
      fdl(2)=dmod(((1.5d-8*dt-5.317d-6)*dt+1236.8530874444d0)*dt+
     +0.8273621196d0,1d0)*dvepi
      fdl(3)=dmod(((8.5d-9*dt-1.0229d-5)*dt+1342.2278264946d0)*dt+
     +0.2590886397d0,1d0)*dvepi
      fdl(4)=dmod(((-9.3d-9*dt-4.452d-7)*dt+99.9973620556d0)*dt+
     +0.9931325648d0,1d0)*dvepi
      fdl(5)=dmod(((4.9d-8*dt+2.4159d-5)*dt+1325.5524094390d0)*dt+
     +0.3748971705d0,1d0)*dvepi
c  vypocet koeficientu casove zavislych clenu
      do 1 j=1,15
      i=no(j)
      p(i)=po(j)+ifix(t*pt(j)/10.)
    1 e(i)=eo(j)+ifix(t*et(j)/10.)
c  nutace
      ddeps=0.
      dpsi=0.
      do 2 n=1,106
      alf=0.
```

```
      i=arg(n)
      do 3 j=1,5
      alf=alf+fdl(j)*(mod(i,10)-5)
    3 i=i/10
      ddeps=ddeps+dfloat(e(n))*cos(alf)
    2 dpsi=dpsi+dfloat(p(n))*sin(alf)
      ddeps=ddeps/drovt/1d4
      dpsi=  dpsi/drovt/1d4
      deps=((dble((1.813e-3*t-5.9e-4)*t)-46.815d0)*dt+84381.448d0)/drovt
     ++ddeps
      dmi=dpsi*dcos(deps)
      dni=dpsi*dsin(deps)
      return
      end
      subroutine longj3(en)
c computation of longper. terms for epoche djne+en
c resulting terms are add to elements in common/elmn/
      implicit real*8 (a-h,o-z)
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     &n1,jp1,jg1,ji1,je1,ja1,jm1
      data rj2,rj3,ae/1082.626d-6,-2.532d-6,6378.137d0/
      dpi=atan2(0d0,-1d0)*2d0
      omega=elemen(ep,en,jp1)
      ri   =elemen(ei,en,ji1)
      e    =elemen(ee,en,je1)
      a    =elemen(ea,en,ja1)
           if(e.eq.0d0) then
           write(66,101)
  101 format(' longper. terms are not included, ecc = 0 !')
                           return
                         endif
      t1=0.5d0
      s=dsin(ri)
      c=dcos(ri)
      eta=sqrt(1d0-e*e)
      p=a*eta**2
      rj=rj3/rj2/p*ae
      p21=t1*rj*(-c/s*e)
      p11=-p21*c+t1*rj*(c*c/s*e-s/e)
      p41=t1*rj*(-s*eta**2)
      p31=-e/eta**2*c/s*p41
      p51=t1*rj*s*eta**3/dpi/e
      co=dcos(omega)
      so=dsin(omega)
      ep(1)=ep(1)+p11*co
      eg(1)=eg(1)+p21*co
      ei(1)=ei(1)+p31*so
      ee(1)=ee(1)+p41*so
      em(1)=em(1)+p51*co
      rod=360d0/dpi
      write(66,102) p11*co*rod,p21*co*rod,p31*so*rod,
     &p41*so*rod,p51*co
  102 format('   longper. terms (j3 only) :'/
     &'   dp :',f10.5/'   dg :',f10.5/'   di :',f10.5/'   de :',
     &f10.5/'   dm :',f10.5/)
      return
      end
```

Appendix 4

Programme PIXPO 4 for Calculation of Pixel Coordinates

```
c program  pixpo4 (inedependet test of pixpo3)  kostelecky 1991
      implicit real*8(a-h,o-z)
      character*80 jmeno
      common/type/ it
      common/utl/ cut
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     &n1,jp1,jg1,ji1,je1,ja1,jm1
      print *,'program pixpo4'
      print *,'give name of set of elements'
      rod=180d0/atan2(0d0,-1d0)
      read(*,'(a)') jmeno
      ijm=index(jmeno,' ')-1
      open(55,file=jmeno(:ijm)//'.dat',status='unknown')
      open(66,file=jmeno(:ijm)//'.lst',status='unknown')
      write(*,7001) jmeno(:ijm)
 7001 format(' output will be in the file:',a,'.lst')
      read(55,*) ilem,lprj3,tskew
       cut=0d0
      it=55
      call inel(ilem)
      call outel(ilem)
      print *,'give time step [sec]:'
      read(*,*) ddtl1
      write(66,202) ddtl1
  202 format (' program pixpo4'/' step in computation [sec]:',f8.2)
      read(55,*,end=2) jnl1,ih,min,sec,aut,but,jdu
      irok=jnl1/10000
      i=jnl1-irok*10000
      nm=i/100
      nd=i-nm*100
      irok=irok+1900
      cut=(aut+but*(djne-jdu))/86400d0
      dtl1=ddtl1/86400d0
      call jayday(nd,nm,irok,jtl1)
      tl11=float(jtl1)+(float(ih)+float(min)/60d0+sec/3600d0)/24d0
        if(lprj3.ne.0) call longj3(tl11-djne)
        lprj3=0
      print *,'tl1',tl1
        call nuta83(tl11)
        write(6,301)
  301   format(' point no. fi(geod.)   lambda    h(m)   date   ',
     &'   time culm.  delta(deg)'/)
    1 read(55,*,end=2) np, rfi, rlam, rrh
      tl1 = tl11
      tkul=0d0
      pfi=rfi/rod
      plam=rlam/rod
        call pixpo44(pfi,plam,rrh,tl1,dtl1,tkul,delta)
        if(tkul.eq.0d0) go to 11
      delt=delta*rod
      jnl=tkul
      call ahms(ih,min,secc,tkul-jnl)
      call calday(iden,mes,irok,jnl)
      write(66,201)np,rfi,rlam,rrh,iden,mes,irok,ih,min,secc,delt
      write(*, 201)np,rfi,rlam,rrh,iden,mes,irok,ih,min,secc,delt
  201 format(' ',i6,1x,2f11.6,f6.1,2x,2(i2,1h.),i4,1x,2(i2,1h:),f5.2,
     &f8.2)
   11 print *,'other data ? [y/n]'
      read(*,'(a)') yn
      if(yn.eq.'y'.or.yn.eq.'y') go to 1
    2 stop
      end
c ***********************************************************************
      subroutine genaka(x,y,z,e,en,h,dzeta,b,el)
      implicit real*8(a-h,o-z)
      x=(en+h+dzeta)*dcos(b)*dcos(el)
      y=(en+h+dzeta)*dcos(b)*dsin(el)
      z=(en*(1d0-e**2)+h+dzeta)*dsin(b)
      return
      end
c ***********************************************************************
      subroutine xyz1(djd,xv)
c inertial geocentric position and velocity of satellite
c call inel and nuta83 before this procedure!
      implicit real*8(a-h,o-z)
      dimension el(9),xv(6),par(9),ddd(4)
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     &n1,jp1,jg1,ji1,je1,ja1,jm1
      common/nu/ dmi,ddd
      dpi=atan2(0d0,-1d0)*2d0
        en=djd-djne
        print *,'en',en
        el(1)=elemen(eg,en,jg1)
        el(2)=elemen(ei,en,ji1)
        el(3)=elemen(ep,en,jp1)
        el(4)=elemen(ea,en,ja1)
        el(5)=elemen(ee,en,je1)
        if(el(5).lt.0d0 ) el(5)=0d0
        n3=0
        a=0d0
        am=elemen(em,en,jm1)
        if(n1.le.0) go to 1
        n2=idint(am)
        if(am.lt.0d0) n2=n2-1
        n3=n1+n2
    1 continue
        am=dmod(am,1d0)*dpi
        el(6)=angle(am-a,dpi)
        el(8)=em(2)*dpi
        call epler(el(6),el(5),el(7),el(9))
        call covel2(-1,el,xv,par)
        return
        end
c ***********************************************************************
      subroutine pixpo44(pfi,plam,rrh,tl1,dtl1,tkul,delta)
c time of culmination and off-nadir angle "delta"
c inputs:
c pfi     - fi geodetic latitude of the pixel [rad]
c plam    - lambda longitude of pixel [rad]
c tl1     - time of first  line (arbitrary but as close as
c                              possible [mjd])
c dtl1    - time of the scan line or arbitrary time interval
c           [fraction of day]
c outputs:
c tkul    - time of culmination
c delta   - off-nadir angle "delta"
c before using this subroutine, read the orbital elements by inel !
c
      implicit real*8(a-h,o-z)
      dimension xv(6),xp(3),xpo(3)
      dimension xt(3),xx(3)
      common/utl/ cut
      pa = 6378137.0
      dii= 1d0/298.257d0
      ea=sqrt(2d0*dii-dii**2)
      en=pa/sqrt(1d0-ea**2*sin(pfi)**2)
      call genaka(xp0,yp0,zp0,ea,en,rrh,0d0,pfi,plam)
      xp(1)=xp0/1d3
```

```
      xp(2)=yp0/1d3
      xp(3)=zp0/1d3
      print *,'xp',xp
      krok=1
 50   call xyz1(tl1,xv)
      call xtt(xv(1),xv(2),xv(3),xt)
      call vrp(xv,xt,xp,tl1,odl1)
      if (krok.eq.1)   tl2=tl1+dtl1
 52   call xyz1(tl2,xv)
      call xtt(xv(1),xv(2),xv(3),xt)
      call vrp(xv,xt,xp,tl1,odl2)
      if((odl1.gt.0d0 .and. odl2.gt.odl1).or.(odl1.lt.0d0 .and.
     & odl2.lt.odl1)) then
         print *,'culmin. was earlier than given time'
         return
         endif
      if(odl1/odl2 .lt. 0d0) go to 51
      odl1=odl2
      tl1=tl2
      tl2=tl2+dtl1
      go to 52
 51   dt=(tl2-tl1)*odl1/(odl1-odl2)
      if(krok.eq.2) go to 53
      dtl2=dtl1/10d0
      tl1=tl1+dt-dtl2
      tl2=tl1+2*dtl2
      krok=2
      go to 50
 53   tkul=tl1+dt
      do 54 i=1,2
 54   xx(i)=xp(i)
      hg=dthg83(tkul+cut)
      dc=dcos(hg)
      ds=dsin(hg)
      xp(1)=xx(1)*dc-xx(2)*ds
      xp(2)=xx(1)*ds+xx(2)*dc
      call xyz1(tkul,xv)
c computation of the off-nadir angle "delta"
      do 8 i=1,3
    8 xpo(i)=xv(i)-xp(i)
      xd1=xv(2)*xpo(3)-xv(3)*xpo(2)
      xd2=xv(3)*xpo(1)-xv(1)*xpo(3)
      xd3=xv(1)*xpo(2)-xv(2)*xpo(1)
      sd=sqrt(xd1**2+xd2**2+xd3**2)
      cd=0
      do 9 i=1,3
    9 cd=cd+xv(i)*xpo(i)
      delta     = atan2(sd,cd)
c sign of delta, doleva -, doprava +
      xmax1=amax1(abs(xd1),abs(xd2),abs(xd3))
      if(xmax1.eq.abs(xd1).and.xd1/xv(4).lt.0d0) delta=-delta
      if(xmax1.eq.abs(xd2).and.xd2/xv(5).lt.0d0) delta=-delta
      if(xmax1.eq.abs(xd3).and.xd3/xv(6).lt.0d0) delta=-delta
      return
      end
c ****************************************************************
      subroutine vesouc(x11,x22,x3)
c vector product and its normalisation
      implicit real*8 (a-h,o-z)
      dimension x11(3),x22(3),x1(3),x2(3),x3(3)
      s22 =0d0
      s11=0d0
      do 1 i=1,3
      s11=s11+x11(i)**2
    1 s22=s22+x22(i)**2
      do 2 i=1,3
      x1(i)=x11(i)/sqrt(s11)
    2 x2(i)=x22(i)/sqrt(s22)
      x3(1)=x1(2)*x2(3)-x1(3)*x2(2)
      x3(2)=x1(3)*x2(1)-x1(1)*x2(3)
      x3(3)=x1(1)*x2(2)-x1(2)*x2(1)
      s3=0d0
      do 3 i=1,3
    3 s3=s3+x3(i)**2
      do 4 i=1,3
      x3(i)=x3(i)/sqrt(s3)
    4 continue
      return
      end
c     ***                                          *******
      subroutine xtt(x1,x2,x3,xt)
c unit vector of the local vertical in satellite point
      implicit real*8 (a-h,o-z)
      dimension xt(3)
       pa=6378.137d0
       dii=1d0/298.257d0
       ea=sqrt(2d0*dii-dii**2)
       ei=sqrt(ea**2/(1d0-ea**2))
       call kanage(x1,x2,x3,ell,be,haa,pa,ps,ei,ea)
      xt(1)=-dcos(be)*dcos(ell)
      xt(2)=-dcos(be)*dsin(ell)
      xt(3)=-dsin(be)
      return
      end
c ****************************************************************
      subroutine vrp(xv,xt,xp,tl1,odl)
c distance of a point from a plane
      implicit real*8 (a-h,o-z)
      dimension xv(6),xt(3),xp(3),xpi(3),xn(3),xb(3),xv1(3)
      hg=dthg83(tl1)
      dc=dcos(hg)
      ds=dsin(hg)
      xpi(1)=xp(1)*dc-xp(2)*ds
      xpi(2)=xp(1)*ds+xp(2)*dc
      xpi(3)=xp(3)
      do 2 i=1,3
    2 xv1(i)=xv(i+3)
      call vesouc(xv1,xt,xn)
      call vesouc(xt,xn,xb)
      s=0d0
      do 1 i=1,3
    1 s=s+xb(i)*(xpi(i)-xv(i))
      odl=s
      return
      end
c
c
      subroutine kanage(x,y,z,ell,be,haa,pa,ps,ei,ea)
      implicit real*8(a-h,o-z)
      ell=datan2(y,x)
      p=dsqrt(x**2+y**2)
      be=datan((z/p)*(1d0+ei**2))
    1 enn=pa/(dsqrt(1d0-(ea**2)*dsin(be)**2))
      ha=p/dcos(be)-enn
      bea=datan((z/p)*((1d0-(enn*ea**2)/(enn+ha))**(-1)))
      ena=pa/(dsqrt(1-(ea**2)*dsin(bea)**2))
      haa=p/dcos(bea)-ena
      if(dabs(haa-ha).lt.3d-4.and.dabs(bea-be).lt.5d-12) go to 2
```

```
      be=bea
      go to 1
    2  be=bea
      return
      end
c
c
      subroutine longj3(en)
c computation of longper. terms for epoch djne+en
c resulting terms are add to elements in common/elmn/
      implicit real*8 (a-h,o-z)
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     &n1,jp1,jg1,ji1,je1,ja1,jm1
      data rj2,rj3,ae/1082.626d-6,-2.532d-6,6378.137d0/
      dpi=atan2(0d0,-1d0)*2d0
      omega=elemen(ep,en,jp1)
      ri   =elemen(ei,en,ji1)
      e    =elemen(ee,en,je1)
      a    =elemen(ea,en,ja1)
           if(e.eq.0d0) then
           write(66,101)
  101 format(' longper. terms are not included, ecc = 0 !')
                              return
                         endif
      t1=0.5d0
      s=dsin(ri)
      c=dcos(ri)
      eta=sqrt(1d0-e*e)
      p=a*eta**2
      rj=rj3/rj2/p*ae
      p21=t1*rj*(-c/s*e)
      p11=-p21*c+t1*rj*(c*c/s*e-s/e)
      p41=t1*rj*(-s*eta**2)
      p31=-e/eta**2*c/s*p41
      p51=t1*rj*s*eta**3/dpi/e
      co=dcos(omega)
      so=dsin(omega)
      ep(1)=ep(1)+p11*co
      eg(1)=eg(1)+p21*co
      ei(1)=ei(1)+p31*so
      ee(1)=ee(1)+p41*so
      em(1)=em(1)+p51*co
      rod=360d0/dpi
      write(66,102) p11*co*rod,p21*co*rod,p31*so*rod,
     &p41*so*rod,p51*co
  102 format(' longper. terms (j3 only) :'/
     &' dp :',f10.5/' dg :',f10.5/' di :',f10.5/' de :',f10.5/
     &' dm :',f10.5/)
      return
      end
c
c
      subroutine ahms(nh,nmin,s,c)
c
c     improved version
c     transformation into hours,minutes and seconds
c    (or into degrees etc. if c is multiplied by 15)
c        input:  c  - fractions of day
c        output: nh - hours
c                nm - minutes
c                s  - seconds (exact to miliseconds)
c
      implicit real*8 (a-h,o-z)
      real*8 s,c
      t=dabs(c*24d0)
      nh=idint(t)
      t=dmod(t,1d0)*60d0
      nmin=idint(t)
      s=dmod(t,1d0)*60d0
      if(dabs(s-6d1).ge.55d-5)go to 1
      nmin=nmin+1
      s=1d-5
      if(nmin.ne.60) go to 1
      nh=nh+1
      nmin=0
    1 if(c.lt.0d0) nh=-nh
      return
      end
      real function angle*8 (b,r)
      implicit real*8(a-h,o-z)
      real*8 b,r
c
c     normalisation of  angle b into interval:
c     (0,h) if r.gt.0 (where h=dabs(r)), or
c     (-h/2,+h/2) if r.lt.0
c
      h=dabs(r)
      a=dmod(b,h)
      if(r)1,1,2
    1 if(dabs(a).gt.0.5d0*h)a=a-dsign(h,a)
c instrukci go to 3 prehodil j.kost.
      go to 3
    2 if (a.lt.0d0) a=a+h
c     goto 3
    3 angle=a
      return
      end
      real function antk*8(v,e)
c
c     computation of the mean anomaly(rad) from:
c     input:  v-true anomaly(rad)
c             e-eccentricity(-)
c
      implicit real*8 (a-h,o-z)
      real*8 v,e
      a=(1d0-e)/(1d0+e)
      b=v
      if(dabs(dabs(b)-3.14159265) .le.1d-5) goto 1
      f=2d0*datan(dsqrt(a)*dtan(0.5d0*v))
      b=f-e*dsin(f)
    1 antk=b
      return
      end
      subroutine calday(nd,nm,ny,jnl)
c     gives gregorian calendar day,month and year cor-
c     responding to jnl=days since nov.17.0 ut 1858
c
      if(jnl.lt.15384)go to 10
      if(jnl.lt.88067)go to 11
   10 nm=0
      nd=0
      ny=0
      go to 13
   11 nseq=jnl-15078
      ny=(4*nseq-1)/1461
      nd=(4*nseq+3-1461*ny)/4
      nm=(5*nd-3)/153
      nd=(5*nd+2-153*nm)/5
```

```
      if(nm.ge.10)go to 12
      nm=nm+3
      ny=ny+1900
      go to 13
   12 nm=nm-9
      ny=ny+1901
   13 return
      end
c
      real function change*8 (x,dy,de,t)
c
c  computation of the real rate of change of an angle
c  input : x - time difference
c          dy- angle difference
c          de- theoretical rate
c          t - angle unit used
c
      implicit real*8 (a-h,o-z)
      real*8 x,dy,de,t
      dx=dabs(x*de)/t
      j=idint(dx)
      dx=dmod(dx,1d0)
      se=dsign(1d0,de)
      if (se.ne.dsign(1d0,x)) dy=-dy
      dy=angle(dy,t)/t
      dxy=dx-dy
      if (dabs(dxy).gt.0.5d0) j=j+idint(dsign(1d0,dxy)+0.5d0)
      dy=dfloat(j)+dy
      change=se*dy*t/x
      return
      end
      subroutine covel2(ilem,el,xv,par)
c
c     corrected variant
c     coordinates and velocity from elements
c      (duboshin 1, p.354 and 370)
c     input:  ilem-type of elements used((see subroutine inel)
c                  (if ilem.eq.3.or.lt.0, short-periodic perturbations
c                  are included)
c             el-array of elements as computed by function elemen.
c     output: xv-array of satellite rectangular coordinates(km)
c               and velocity(km/s)
c             par(1-3)-directional cosines
c             par(4-6)-first derivatives of cosines
c             par(7-9)-second derivatives of cosines
c
      implicit real*8 (a-h,o-z)
      dimension el(9),xv(6),par(9)
      data  rz, gm/6.37814d3,398601.3d0/
      if(ilem.eq.3) goto 3
      if(ilem.lt.0) goto 4
      dr=0d0
      di=0d0
      dl=0d0
      dm=0d0
      goto 10
    4 call short(dr,di,dl,dm,el)
      goto 10
    3 call elas (dr,di,dl,dm,el)
c     write(6,20) dr,di,dl,dm
c   20 format(1h ,80x,f8.3,3d11.3)
   10 sv=dsin(el(7))
      cv=dcos(el(7))
      au=el(7)+el(3)
      au=au+dl
      su=dsin(au)
      cu=dcos(au)
      el(1)=el(1)+dm
      sg=dsin(el(1))
      cg=dcos(el(1))
      el(2)=el(2)+di
      si=dsin(el(2))
      ci=dcos(el(2))
      pp=el(4)*(1d0-el(5)*el(5))
      cx=dsqrt(gm/pp)
      c2=1d0+el(5)*cv
      r=pp/c2
      r=r+dr
      par(1)=cu*cg-su*sg*ci
      par(2)=cu*sg+su*cg*ci
      par(3)=su*si
      par(4)=-su*cg-cu*sg*ci
      par(5)=-su*sg+cu*cg*ci
      par(6)=cu*si
      par(7)=sg*si
      par(8)=-cg*si
      par(9)=ci
      do 1 i=1,3
      xv(i)=r*par(i)
      xv(i+3)=cx*(par(i)*el(5)*sv+par(i+3)*c2)
    1 continue
      return
      end
      subroutine deltae(en,a,e,dn,da,de)
c
c     computation of missing parameters for subroutine inel (supposing
c     only stationary atmosphere's drag effect is involved)
c     input:  en-mean daily motion(rev/day)
c             a-semimajor axis(km)
c             e-eccentricity
c             dn-(1/2.dn) change of mean motion(rev/day2)
c     output: da-change of semimajor axis(km/day)
c             de-change of eccentricity(-/day)
c
      implicit real*8(a-h,o-z)
      real*8 en,a,e,dn,da,de
      de=0d0
      x=-4d0*dn/(3d0*en)
      da=x*a
      if(da.ge.0d0)return
      de=(1d0-e)*x
      return
      end
      subroutine detor(e,x)
c
c     corrected variant
c     elements from coordinates and velocity
c      (duboshin 1, p.292-296)
c     input: x-array of satellite coordinates and velocity(km,km/s).
c     subroutines: angle,antk
c     output: osculating elements:
c             e(1)-ascending node(rad)
c             e(2)-inclination(rad)
c             e(3)-argument of perigee(rad)
c             e(4)-semimajor axis(km)
c             e(5)-eccentricity(-)
c             e(6)-mean anomaly(rad)
c             e(7)-true anomaly(rad)
```

```
c            e(8)-mean daily motion(rad/day)
c            e(9)-(not computed)
c
      implicit real*8(a-h,o-z)
      dimension e(9),x(6),aj(21)
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
      data  rz,gm /6.37814d3,398601.3d0/
      data aj(2),aj(3),aj(4),aj(5)/1.082639d-3,-2.546d-6,-1.649d-6,
     *        -0.210d-6/
      r=dsqrt(x(1)*x(1)+x(2)*x(2)+x(3)*x(3))
      v=dsqrt(x(4)*x(4)+x(5)*x(5)+x(6)*x(6))
      cm=-gm/r
      c1=x(2)*x(6)-x(3)*x(5)
      c2=x(3)*x(4)-x(1)*x(6)
      c3=x(1)*x(5)-x(2)*x(4)
      c=dsqrt(c1*c1+c2*c2+c3*c3)
      f1=cm*x(1)+c3*x(5)-c2*x(6)
      f2=cm*x(2)+c1*x(6)-c3*x(4)
      f3=cm*x(3)+c2*x(4)-c1*x(5)
      f=dsqrt(f1*f1+f2*f2+f3*f3)
      e(1)=datan2(c1,-c2)
      e(2)=datan2(dsqrt(c1*c1+c2*c2),c3)
      p=c*c/gm
      e(5)=f/gm
      a=2d0/r-v*v/gm
      e(4)=1d0/a
      ci=dcos(e(2))
      sg=dsin(e(1))
      cg=dcos(e(1))
      if (dabs(ci).le.1d-15) goto 5
      sp=(-f1*sg+f2*cg)/ci
      cp= f1*cg+f2*sg
      su=(-x(1)*sg+x(2)*cg)/ci
      cu= x(1)*cg+x(2)*sg
      goto 6
    5 sp=f3
      su=x(3)
      if (dabs(cg).le.1d-2) goto 7
      cp=f1/cg
      cu=x(1)/cg
      goto 6
    7 cp=f2/sg
      cu=x(2)/sg
    6 u=datan2(su,cu)
      e(3)=datan2(sp,cp)
      e (7)=angle(u-e(3),dpi)
      e(6)=antk(e (7),e(5))
      aa=e(4)/rz
      aa=aa*aa
      dj2=1.5d0*aj(2)/aa
      an=dsqrt(gm/e(4)**3d0)*864d2
      e(8)=an*(1d0+dj2)
      return
      end
      subroutine elas(dr,di,dl,dm,el)
c     computation of shortperiodic perturbations caused by j2
c     (see osnovy teorii poleta kosmitcheskich apparatov,
c      moskva 1972, pp.347-8)
c     input:  el-array of elements as computed by function elemen.
c     output: dr-perturbation of the radius -vector(km)
c             di-perturbation of the inclination(rad)
c             dl-perturbation of the argument of latitude(rad)
c             dm-perturbation of the ascending node(rad)
c
      implicit real*8(a-h,o-z)
      real*8 dr,di,dl,dm
      dimension el(9)
      aj=66054.6d0
      au=el(7)+el(3)
      sp=dsin(el(3))
      cp=dcos(el(3))
      si=dsin(el(2))
      ci=dcos(el(2))
      s2i=dsin(2d0*el(2))
      aa=el(4)
      ae=el(5)
      su=dsin(au)
      cu=dcos(au)
      s2u=dsin(2.d0*au)
      c2u=dcos(2.d0*au)
      sv=dsin(el(7))
      cv=dcos(el(7))
      a=el(3)+2d0*el(7)
      sp2v=dsin(a)
      cp2v=dcos(a)
      a=2d0*el(3)+el(7)
      s2pv=dsin(a)
      c2pv=dcos(a)
      a=2d0*el(3)+3d0*el(7)
      s2p3v=dsin(a)
      c2p3v=dcos(a)
      p=aa*(1.d0-ae*ae)
      p=p*p
      a=c2u+ae*c2pv+ae*c2p3v/3d0
      a=a-(1d0+4d0*ae*cp/3d0)
      di=aj*s2i*a/p/4.d0
      a=-0.5*su+ae*(sv-0.5d0*s2pv-s2p3v/6d0)
      a=a+4d0*ae*sp/3d0
      dm=-aj*ci*a/p
      a=cu-ae*cv/2.d0-ae*c2pv/2d0-ae*cp2v/2d0
      a=a-(1d0-1.5d0*ae*cp)
      b=si*si
      c=a/(1.d0+ae*cv)+b*(c2u/6d0-2d0*cu/3d0+0.5d0)
      dr=aj*c/aa
      a=(-2.d0+4.d0*b/3.d0)*su+(-0.5d0+7.d0*b/12d0)*s2u+ae*(-2d0*b*sv+
     1(-1.5d0+5d0*b/6d0)*s2pv+(-1d0+b)*s2p3v/6d0+(-0.5d0+b/3d0)*sp2v)
      a=a+(5d0-14d0*b)*ae*sp/6d0
      dl=aj*a/p
      return
      end
      real function elemen*8(a,e,i)
c
c     computation of orbital element from time polynomials.
c     input:  a-array of the polynomial coefficients
c             e-time elapsed from epoch(days)
c             i-limiting power of e
c
      implicit real*8(a-h,o-z)
      dimension a(8)
      real*8 e
      b=a(1)
      k=2
    3 if(i-k)1,2,2
    2 b=b+a(k)*e**(k-1)
      k=k+1
      goto 3
    1 continue
      elemen=b
```

```
      return
      end
cc
      subroutine epler (em,ee,av,ae)
c
c     solution of the kepler equation
c     input:  em-mean anomaly (rad)
c             ee-eccentricity (-)
c     output: av-true anomaly(rad)
c             ae-eccentric anomaly(rad)
c
      implicit real*8 (a-h,o-z)
      real*8 em,ee,av,ae
      x=em
      ae=x
      if(dabs(dabs(x)-3.14159265) .le.1d-5) goto 1
    2 x1=x
      x=em+ee*dsin(x)
      if(dabs(x1-x) .gt. 1.e-10) goto 2
      ae=x
      a=(1d0+ee)/(1d0-ee)
      x=2d0*datan(dsqrt(a)*dtan(0.5d0*x))
    1 av=x
      return
      end
      subroutine inel(i)
c
c     input of elements in different codes (improved in 1980)
c     input:i-type of elements:
c             1-injection point parameters
c             2-nasa two-line elements
c             3-orbita code
c             4-five-line code
c             5-sao telex code
c             6-zipsat code
c             7-king-hele's code
c             8-orbit from position and velocity
c          itype- input device specification(for standard input from car
c     common:elmn,type
c     subroutines:angle,antk,deltae,jayday,kozai,sapla,detor
c     output: ep-array of coefficients for arg.of perigee(rad)-tilljp1
c             eg-array of coefficients for asc. node(rad)-till jg1
c             ei-array of coefficients for inclination (rad)-till ji1
c             ee-array of coefficients for eccentricity(-)-till je1
c             ea-array of coefficients for semimajor axis(km)-till ja1
c             em-array of coefficients for mean anomaly(rev)-till jm1
c             djne-epoch of elements(mjd)
c             sate-name of the satellite
c     note 1: end of data set could be identified as jp1=0(generated
c                    by inel)
c     note 2: in sao format, variable ep(3) is used for storage of time
c             interval till next epoch of elements(this is used
c             in program sunrad2 etc.)
c
      implicit real*8(a-h,o-z)
      character*8 sate
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     1n1,jp1,jg1,ji1,je1,ja1,jm1
      common/type/ itype
      dimension aj(21),e(9),x(6)
      data  dhg,hg1/7.292115147d-5,6.300387487d0/
      data  rz,dmi/6.37814d3,398601.3d0/
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
      data aj(2),aj(3),aj(4),aj(5)/1.082639d-3,-2.546d-6,-1.649d-6,
     *        -0.210d-6/
      aj(1)=1.5*aj(2)
      it=itype
      if(itype.lt.8.or.itype.gt.100) it=55
      goto (1,2,3,4,5,6,7,8,9),i
c
c     injection point
c
    1 read(it,101,end=50) jne,a,b,c,sl,fi,ei(1),hp,ha,hn,j,sate
  101 format(i5,2f2.0,f5.3,3f7.2,3f8.1,i2,11x,a8)
      if(jne.eq.0) goto 50
      djne=(a*36d2+b*6d1+c)/864d2+jne
      ea(1)=(ha+hp)*0.5d0+rz
      ee(1)=(ha-hp)/(2d0*ea(1))
      a=ei(1)*cdr
      si=dsin(a)
      ci=dcos(a)
      sl=sl*cdr
      fi=fi*cdr
      if(dabs(si).ge.1d-5) goto   21
      ep(1)=dsin(sl)
      eg(1)=dsign(1d0,ci)*dsin(sl)
      go to 22
   21 ep(1)=dsin(fi)/si
      eg(1)=dtan(fi)*ci/si
      if(dabs(ep(1)).le.1d0) goto 22
c  wrong injection data-inclination is set equal to injection latitude
      a=dabs(fi)
      si=dsin(a)
      ci=dcos(a)
      ep(1)=dsign(1d0,ep(1))
      eg(1)=dsign(1d0,eg(1))
   22 ep(1)=dasin(ep(1))
      eg(1)=dasin(eg(1))
      em(1)=0d0
      c=ea(1)/rz
      b=1d0-aj(1)/(c*c)*(1d0-ee(1)*ee(1))**(-1.5d0)*(1d0-1.5d0*a*a)
      em(2)=dsqrt(b*dmi/ea(1)**3d0)*864d2/dpi
      if(j.eq.0) goto 23
      ep(1)=dpi*0.5d0-ep(1)
      eg(1)=dpi*0.5d0-eg(1)
   23 call sapla(djne,hg)
      eg(1)=sl-eg(1)+hg
      if(dabs(hn).le.1d0) go to 24
      b=rz+dabs(hn)
      c=ea(1)*(1d0-ee(1)*ee(1))/b-1d0
      if(ee(1).le.1d-6) go to 24
      b=c/ee(1)
      if(dabs(b).lt.1d0) goto 25
c  wrong injection data - injection point is supposed to be apogee
      em(1)=0.5d0
      ep(1)=ep(1)-em(1)*dpi
      go to 24
   25 em(1)=dsign(1d0,hn)*dacos(b)
      ep(1)=ep(1)-em(1)
      em(1)=antk(em(1),ee(1))/dpi
   24 ei(1)=a/cdr
      ep(1)=ep(1)/cdr
      eg(1)=eg(1)/cdr
      ep(2)=0d0
      eg(2)=0d0
      jp1=2
      jg1=2
      ji1=1
```

```
      ja1=1
      je1=1
      jm1=2
      n1=0
      goto 200
c
c nasa two-line elements
c
    2 read(it,102,end=50) sate,ny,day,em(3),em(4),nx,ei(1),eg(1),ee(1),
     *ep(1),em(1),em(2),n1
  102 format(9x,a8,1x,i2,f12.8,1x,f10.8,1x,f6.5,i2/
     18x,f8.4,f9.4,f8.7,2f9.4,f12.8,i5)
      if (ny.eq.0) goto 50
      call kozai(ei(1)*cdr,ee(1),em(2),ea(1),eg(2),ep(2))
      ny=ny+1900
      call jayday(0,1,ny,jne)
      djne=jne+day
      em(1)=em(1)/36d1
      em(3)=em(3)*0.5d0
      em(4)=em(4)*1d1**nx/6d0
      eg(2)=eg(2)*em(2)*36d1
      ep(2)=ep(2)*em(2)*36d1
      call deltae(em(2),ea(1),ee(1),em(3),ea(2),ee(2))
      jp1=2
      jg1=2
      ji1=1
      ja1=2
      je1=2
      jm1=4
      goto 200
c
c  orbita
c
    3 read(it,103,end=50) n1,jne,a,b,c,em(2),ea(1),ee(1),ei(1),eg(1),
     *ep(1),sate,em(3),ea(2),ee(2),ei(2),eg(2),ep(2)
  103 format(2i5,2f2.0,f5.3,f11.7,f9.3,f6.6,f7.4,2f8.4,7x,a5/5x,6f11.7)
      if(jne.eq.0) goto 50
      djne=(a*36d2+b*6d1+c)/864d2+jne-125d-3
      if(dabs(ep(2)).gt.1d-5.or.dabs(eg(2)).gt.1d-5) goto 13
   56 ci=dcos(ei(1)*cdr)
      aa=ea(1)*(1d0-ee(1)*ee(1))
      aa=6378.15d0/aa
      aa=1.62405d-3*aa*aa*36d1
      eg(2)=-aa*ci
      ep(2)=aa*(2.5d0*ci*ci-0.5d0)
   13 if(eg(1).lt.5d2) goto 14
      call sapla(djne,hg)
      eg(1)=eg(1)+hg/cdr-5d2
   14 em(2)=em(2)/144d1
      em(3)=em(3)/144d1
      a=em(2)-0.5d0*em(3)
      b=1/a
      c=-em(3)/(2d0*a**3d0)
      ep(3)=ep(2)*c
      ep(2)=ep(2)*b
      eg(3)=eg(2)*c
      eg(2)=eg(2)*b
      ei(2)=ei(2)*b
      ee(2)=ee(2)*b
      ea(2)=ea(2)*b
      em(4)=em(3)*em(3)/(2d0*a**5d0)
      em(3)=c
      em(2)=b
      em(1)=0d0
      if(dabs(ea(2)).lt.1d-5) call deltae(em(2),ea(1),ee(1),em(3),
     1ea(2),ee(2))
      jp1=3
      jg1=3
      ji1=2
      ja1=2
      je1=2
      jm1=4
      goto 200
c
c five-line
c
    4 read(it,104,end=50) sate,djne,em(1),eg(1),ep(1),ee(1),ei(1),em(2),
     *em(3),eg(2),ep(2),jdte,idte,ea(1)
  104 format(a8,g14.0,5g8.0/2g11.0,2g8.0,i6,i2,g10.0)
      if (djne.lt.1d1) goto 50
      ea(1)=ea(1)*rz
      em(1)=em(1)/36d1
      call deltae(em(2),ea(1),ee(1),em(3),ea(2),ee(2))
      jp1=2
      jg1=2
      ji1=1
      ja1=2
      je1=2
      jm1=3
      n1=0
      goto 200
c
c  sao telex code (changed in 1980 and 1985 = inclination format f15.10)
c
    5 read(it,*,end=50) sate,djne,ep(1),ep(2),eg(1),eg(2),ei(1),ei(2),
     *ee(1),ee(2),ep(3),em(1),em(2),em(3),em(4)
  105 format(a8,1x,f15.8,f15.10,f13.10,f15.10,f13.10/7x,f15.10,4f13.10/
     * 8x,f13.10,f14.10,1x,d11.4,1x,d11.4)
      if (djne.lt.1d1) goto 50
      call kozai(ei(1)*cdr,ee(1),em(2),ea(1),a,b)
      call  deltae(em(2),ea(1),ee(1),em(3),ea(2),a)
      jp1=2
      jg1=2
      ji1=2
      ja1=2
      je1=2
      jm1=4
      n1=0
      go to 200
c
c zipsat
c
    6 read(it,106,end=50) ny,day,ei(1),ep(1),em(1),em(2),em(3),nx,ee(1),
     *eg(1),sate
  106 format(6x,i2,f10.6,1x,3f7.7,f10.7,f6.5,i1,f8.7,f7.7,a8)
      if (ny.eq.0) goto 50
      call kozai(ei(1)*dpi,ee(1),em(2),ea(1),eg(2),ep(2))
      ny=ny+1900
      call jayday(0,1,ny,jne)
      djne=jne+day
      ep(1)=(ep(1)+0.25)*360d0
      ei(1)=ei(1)*360d0
      em(3)=em(3)*1d1**(-nx-2)
      eg(1)=eg(1)*360d0
      eg(2)=eg(2)*em(2)*360d0
      ep(2)=ep(2)*em(2)*360d0
      call deltae(em(2),ea(1),ee(1),em(3),ea(2),ee(2))
      jp1=2
```

```
      jg1=2
      ji1=1
      ja1=2
      je1=2
      jm1=3
      n1=0
      go to 200
c
c  king-hele
c
    7 read(it,107,end=50) n1,jne,ea(1),ee(1),ei(1),eg(1),ep(1),em(1),
     *em(2),em(3),em(4),sate
  107 format(2i5,f8.3,f5.5,f7.4,3f6.3,f8.4,2f5.5,9x,a5)
      if(jne.eq.0) goto 50
      djne=dfloat(jne)
      ep(2)=0d0
      eg(2)=0d0
      em(1)=em(1)/36d1
      em(2)=em(2)/36d1
      em(3)=em(3)/36d1
      em(4)=em(4)/36d1
      call deltae(em(2),ea(1),ee(1),em(3),ea(2),ee(2))
      jp1=2
      jg1=2
      ji1=1
      ja1=2
      je1=2
      jm1=4
      go to 200
c
c  orbit from position and velocity
c
    8 read(it,108,end=50) n1,jne,a,b,c,j,ax,ay,az,vx,vy,vz,sate,
     * em(2),em(3)
  108 format(2i5,2f2.0,f5.3,i1,6f9.3,1x,a5/5x,2f10.7)
      if(jne.eq.0) goto 50
      djne=(a*36d2+b*6d1+c)/864d2+jne-125d-3
      if (j) 51,51,52
   51 call sapla(djne,hg)
      shg=dsin(hg)
      chg=dcos(hg)
      x(1)=ax*chg-ay*shg
      x(2)=ax*shg+ay*chg
      x(3)=az
      x(4)=vx*chg-vy*shg-dhg*x(2)
      x(5)=vx*shg+vy*chg+dhg*x(1)
      x(6)=vz
      goto 53
   52 x(1)=ax
      x(2)=ay
      x(3)=az
      x(4)=vx
      x(5)=vy
      x(6)=vz
   53 call detor(e,x)
      ep(1)=e(3)/cdr
      ep(2)=0d0
      eg(1)=e(1)/cdr
      eg(2)=0d0
      ei(1)=e(2)/cdr
      ee(1)=e(5)
      ea(1)=e(4)
      if (em(2).le.1d-1) goto 55
      ei(2)=0d0
      i=3
      goto 56
   55 em(1)=e(6)/dpi
      em(2)=e(8)/dpi
      jp1=2
      jg1=2
      ji1=1
      je1=1
      ja1=1
      jm1=2
      goto 200
c
    9 continue
      goto 200
c
  200 if(dabs(ep(2)).gt.1d-5.or .dabs(eg(2)).gt.1d-5) goto 204
      call kozai(ei(1)*cdr,ee(1),em(2),a,b,c)
      if(ea(1).lt.1d0)ea(1)=a
      ep(2)=c*em(2)*36d1
      eg(2)=b*em(2)*36d1
  204 eg(1)=angle(eg(1),360d0)
      ep(1)=angle(ep(1),360d0)
      do 201 k=1,jp1
  201 ep(k)=ep(k)*cdr
      do 202 k=1,jg1
  202 eg(k)=eg(k)*cdr
      do 203 k=1,ji1
  203 ei(k)=ei(k)*cdr
      return
   50 jp1=0
      return
      end
      subroutine jayday(nd,nm,ny,jnl)
c
c     gives jnl=days since nov.1g.0 ut 1858 corres-
c     ponding to gregorian calendar month,day and year.
c
      if(ny.lt.1901)go to 10
      if(ny.lt.2100)go to 11
   10 jnl=0
      go to 14
   11 mm=nm
      my=ny-1900
      if(mm.gt.2)go to 12
      mm=mm+9
      my=my-1
      goto 13
   12 mm=mm-3
   13 jnl=15078+(1461*my)/4+(153*mm+2)/5+nd
   14 return
      end
      subroutine kozai(angr,ecc,aprm1,cp,grr,grp)
c
c     computation of missing parameters for inel (sao spec.rep.no.30)
c     input:  angr-inclination(rad)
c             ecc-eccentricity(-)
c             aprm1-mean daily motion(rev/day)
c     output: cp-semimajor axis(km)
c             grr-rate of change of ascension node(rev/rev)
c             grp-rate of change of argument of perigee(rev/rev)
c
      implicit real*8(a-h,o-z)
      a=aprm1*aprm1
      f=(0.7537138d+14/a)**(1d0/3d0)
```

```
      x=ecc*ecc
      y=1d0-x
      aa=dsqrt(y)
      ab=f*y*f*y
      ac=0.6606372d+5/ab
      sngr=dsin(angr)
      cngr=dcos(angr)
      ae=sngr*sngr
      w=0.7537138d+14*(1d0-ac*aa*(1d0-1.5d0*ae))
      r=w/a
   10 t=f*f
      u=3d0*t
      v=f*t
      dr=r-v
      f=f+dr/u
      if(dabs(dr).gt.0.5d-06*r)go to 10
      cp=f
      ap=ae*ae
      ah=cngr*cngr
      ai=ah*ah
      aj=0.119392d+11/(ab*ab)
      grr=-(ac*cngr*(1d0+ac*(1.5+x/6d0-2d0*aa-ae*(5d0/3d0-5d0*x/24d0
     1-3d0*aa)))+aj*cngr*(6d0/7d0-3d0*ae/2d0)*(1d0+3d0*x/2d0))
      grp=ac*(2d0-2.5*ae)*(1d0+ac*(2d0+x/2d0-2d0*aa-ae*(43d0/24d0
     1-x/48d0-3d0*aa)))-5d0*ac*ac*x*ai/12d0+aj*(12d0/7d0-93d0*ae/14d0
     2+21d0*ap/4d0+x*(27d0/14d0-189d0*ae/28d0+81d0*ap/16d0))
      return
      end
c
      subroutine longps(nn,ai,eo,an,b,otm,otv,p)
c
c     computation of longperiodic and secular perturbations from j2-j5
c     with basic period (see se1/i,p.130).
c     input:  nn-limiting value of jn   (max.5)
c             ai-inclination (rad)
c             eo-eccentricity (-)
c             an-mean daily motion (rev/d)
c             b -semimajor axis (km)
c     output: otm-secular change of perigee(rad/day)
c             otv-secular change of ascending node(rad/day)
c             p(1)-amplitude of longperiodic changes of arg.perigee(rad)
c             p(2)-amplitude of ascending node (rad)
c             p(3)-amplitude of inclination(rad)
c             p(4)-amplitude of eccentricity(-)
c             p(5)-amplitude of mean anomaly at epoch(rad)
c
      implicit real*8 (a-h,o-z)
      dimension c(10),e(10),h(10),g(10),p(15),q(10),a(10),aj(21)
      data aj(2),aj(3),aj(4),aj(5)/1.082639d-3,-2.546d-6,-1.649d-6,
     *        -0.210d-6/
      data  rz,dmi/6.37814d3,398601.3d0/
      n=nn
      aa=b/rz
      su3=float(n)/2
      n=n/2-1
      co=dcos(ai)
      c(1)=co*co
      e(1)=eo*eo
      if(n)115,115,116
  116 do 110 j=1,n
      c(j+1)=c(j)*c(1)
      e(j+1)=e(j)*e(1)
  110 continue
  115 ee=1.-e(1)
      et=dsqrt(ee)
      si=dsin(ai)
      s2=si*si
      c5=1.-5.*c(1)
      c9=8.+c5
      g2=3./(-128.)*(2.*c5*(5.+43.*c(1))+(25.-126.*c(1)+145.*c(2))*e(1)
     1-24.*c5*(1.-3.*c(1))*et)
      h2=3./(-32.)*(4.*(1.-10*c(1))-c9*e(1)+12.*(1.-3.*c(1))*et)
      n=n+1
      go to (102,103),n
  103 h(2)=15./32.*(3.-7.*c(1))*(2.+3.*e(1))
      g(2)=15./128.*(12.-144.*c(1)+196.*c(2)+e(1)*(9.-126.*c(1)+189.*
     1c(2)))
  102 h(1)=1.5
      g(1)=-0.75*c5
      po=aa*ee
      su1=0.
      su2=0.
      p2=po*po
      pj=1.
      do 101 j=1,n
      pj=pj*p2
      ajp=aj(2*j)/pj
      su1=su1+ajp*g(j)
      su2=su2+ajp*h(j)
  101 continue
      pj=p2*p2
      ajp=aj(2)*aj(2)/pj
      otm=an*(su1+ajp*g2)*6.28318531
      otv=(-an*co)*(su2+ajp*h2)*6.28318531
      su3=su3-n
      su3=dabs(su3)-0.1
      if(su3)120,120,121
  120 n=n-1
  121 go to (113,114),n
  114 g(2)=1.-14.*c(1)+21.*c(2)
      h(2)=28.-84.*c(1)
      pj=5./32.
      p(2)=(4.+3.*e(1))*pj
      q(2)=pj*(4.+25.*e(1)+6.*e(2))
      a(2)=pj*g(2)*(4.+9.*e(1))
  113 g(1)=c5
      h(1)=10.*c5*c5
      p(1)=0.5
      q(1)=0.5
      a(1)=0.5*c5
      su1=0.
      su2=0.
      su3=0.
      su4=0.
      pj=1./po
      do 111 j=1,n
      jj=2*j+1
      pj=pj*p2
      ajp=aj(jj)/pj
      ajq=ajp*q(j)
      ajp=ajp*p(j)
      su1=su1+ajp*g(j)
      su2=su2+ajp*(c9*g(j)-s2*c5*h(j))
      su3=su3+ajq*g(j)
      su4=su4+ajp*a(j)
  111 continue
      pj=aj(2)*c5
      if(dabs(pj)-0.1e-9)127,127,128
```

```
  127 pj=1.0e15*dsign(1.d0,si)/dsign(1.d0,pj)
      go to 129
  128 pj=si/pj
  129 p(4)=pj*(-su1)*ee
      p(1)=pj*su3/eo
      p(5)=pj*su4*et*et*et/(eo*6.28318531)
      if(dabs(si)-0.1e-6)122,122,123
  122 pj=1.0e15*dsign(1.d0,co)
      go to 124
  123 pj=eo*co/si
  124 p(3)=pj*(-p(4))/et
      if(dabs(c5)-0.1e-6)125,125,126
  125 p(2)=1.0e18*dsign(1.d0,pj)*dsign(1.d0,su2)/dsign(1.d0,aj(2))
      goto 130
  126 p(2)=pj*su2/(aj(2)*c5*c5)
  130 p(1)=co*(-p(2))-p(1)
      do 100 i=6,15
  100 p(i)=0d0
      return
      end
      subroutine nhms(nh,nm,ns,c,i)
c
c     transformation from fractions of day into hours,minutes
c     and seconds.
c     input : c-fractions of day
c             i-if equal to 1,result is rounded to integral seconds,
c              -if equal to -1,result is rounded to integral minutes.
c     output : nh-hours
c              nm-minutes
c              ns-seconds(rounded)
c
      implicit real*8(a-h,o-z)
      ns=0
        t=dabs(c*24d0)
      nh=idint(t)
      t=dmod(t,1d0)*6d1
      if(i)1,1,2
    1 nm=idint(t+0.5d0)
      go to 4
    2 nm=idint(t)
      s=dmod(t,1d0)*6d1
      ns=idint(s+0.5d0)
      if(ns.lt.60) go to 3
      nm=nm+1
      ns=0
    4 if(nm.lt.60) go to 3
      nh=nh+1
      nm=0
    3 if(c.lt.0d0) nh=-nh
      return
      end
      subroutine orbnas (ilem)
c
c     transformation of input elements from code orbita (ilim=3)
c     into nasa two-line code (ilem=2)
c     if ilem.lt.o punch the resulting elements
c     attention - value of ilem is changed during execution
      implicit real*8(a-h,o-z)
      character*8 sate
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,
     &sate,n1,jp,jg,ji,je,ja,jm
      data c20,rz/-1082.6d-6,6378.14d0/
      dpi=datan2(0d0,-1d0)*2d0
      j23r=-7.173247d0
      rod=36d1/dpi
      em(1)=antk(-ep(1),ee(1))*rod
      alfa=1.5d0*(rz/ea(1))**2*((2.5d0*dcos(ei(1))**2-0.5d0)/
     &((1d0+ee(1)*dcos(ep(1)))**2*dsqrt(1d0-ee(1)**2))
     &+(1d0+ee(1)*dcos(ep(1)))**3/(1d0-ee(1)**2)**3)
      t0=1440d0/em(2)/(1d0+c20*alfa)
      tome=t0*(1d0+1.5d0*c20*(rz/ea(1))**2*((1d0+ee(1)*
     &dcos(-ep(1)))/(1d0-ee(1)**2))**3)
      em(2)=1440d0/tome
      ee(1)=ee(1)+j23r/ea(1)*dsin(ei(1))*dsin(ep(1))
      ei(1)=ei(1)*rod
      eg(1)=eg(1)*rod
      ep(1)=ep(1)*rod
      jnem=idint(djne)
      call calday(nd,nm,nrok,jnem)
      call jayday( 0, 1,nrok,jnem1)
      porden=djne-jnem1
      ny=nrok-1900
      write(66,200) sate,ny,porden,em(3),ei(1),eg(1),ee(1),ep(1),
     &em(1),em(2)
  200 format(' ',' elements in nasa two-line code',
     &//9x,a8,1x,i2,f12.8,1x,f10.8,/8x,f8.4,f9.4,f8.7,
     &2f9.4,f12.8)
      if (ilem.lt.0)
     *write(7,201) sate,ny,porden,em(3),ei(1),eg(1),ee(1),ep(1),
     &em(1),em(2)
  201 format(9x,a8,1x,i2,f12.8,1x,f10.8,1x/8x,f8.4,f9.4,f8.7,
     &2f9.4,f12.8)
      ei(1)=ei(1)/rod
      eg(1)=eg(1)/rod
      ep(1)=ep(1)/rod
      em(1)=em(1)/36d1
      ilem=2
      call outel(ilem)
      return
      end
      subroutine outel(i)
c
c     print of elements obtained by inel.
c     angular elements are printed in (degrees) or (degrees/day),
c     ea in (km), em in (rev) or (rev/day).
c      input: i-type of elements
c     common:elmn
c      subroutines: ahms,calday
c      output: (none)
c
      implicit real*8(a-h,o-z)
      character*8 sate
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     1n1,jp1,jg1,ji1,je1,ja1,jm1
      character fff
      dimension ap(3),ag(3),ai(2)
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
c
      fff=char(12)
      write(66,100) sate
  100 format('  satellite : ',a8/'   source of elements : ')
      goto (1,2,3,4,5,6,7,8,9,10),i
    1 write(66,11)
   11 format(' ',27x,'injection point'/)
      goto 200
    2 write(66,12)
   12 format(' ',27x,'nasa two-line elements'/)
      go to 200
```

```
    3 write(66,13)
   13 format(' ',27x,'orbita'/)
      em(1)=0d0
      go to 200
    4 write(66,14)
   14 format(' ',27x,'five-line code'/)
      go to 200
    5 write(66,15)
   15 format(' ',27x,'sao telex code'/)
      go to 200
    6 write(66,16)
   16 format(' ',27x,'zipsat'/)
      go to 200
    7 write(66,17)
   17 format(' ',27x,'king-hele'/)
      goto 200
    8 write(66,18)
   18 format(' ',27x,'position and velocity'/)
      goto 200
    9 continue
      goto 200
   10 write(66,20)
   20 format(' ',27x,'program prior'/)
      goto 200
  200 jne=idint(djne)
      ajne=dmod(djne,1d0)
      call calday(md,mm,my,jne)
      call ahms(nh,nm,s,ajne)
      do 201 k=1,jp1
  201 ap(k)=ep(k)/cdr
      do 202 k=1,jg1
  202 ag(k)=eg(k)/cdr
      do 203 k=1,ji1
  203 ai(k)=ei(k)/cdr
c
      if(n1.gt.0) write(66,204)n1
  204 format(' ',2x,'orbit : ',i11)
      write(66,205)jne,ajne,md,mm,my,nh,nm,s
  205 format(' ',2x,'epoch : ',6x,i5,3x,f8.6,2x,2(i2,'.'),i4,
     12x,2(i2,':'),f5.2)
      write(66,206)(ap(k),k=1,jp1)
  206 format(' ' ,2x,'perig : ',2f11.6,2d14.6)
      write(66,207)(ag(k),k=1,jg1)
  207 format(' ',2x,'rnode : ',2f11.6,2d14.6)
      write(66,208)(ai(k),k=1,ji1)
  208 format(' ',2x,'incln : ',2f11.6,2d14.6)
      write(66,209)(ee(k),k=1,je1)
  209 format(' ',2x,'eccen : ',2f11.6,2d14.6)
      write(66,210)(ea(k),k=1,ja1)
  210 format(' ',2x,'saxis : ',2f11.3,2d14.6)
      write(66,211)(em(k),k=1,jm1)
  211 format(' ',2x,'manom : ',2f11.6,2d14.6)
      if(i.eq.3) write(66,212)
  212 format(' ',12x,'(mean nodal motion is given)')
      write(66,213)
  213 format(' ')
      return
      end
c
      subroutine posic2(enl,i,k,n,x,u,n3)
c
c     computation of geocentric coordinates for given elapsed time
c      input:enl-elapsed  time from epoch(days)
c            i-type of elements(1-7).
c            (if.le.0.or.eq.3,shortperiodic pert. are included)
c     for pc are not included!
c           k-if.gt.0, tesseral pert. are included
c           n-index for perturbation array dal
c      common: elmn,el
c      subroutines:angle,antk,covel2,elemen,epler
c     output for covel2:array of actual values of elements:
c             el(1)-ascending node(rad)
c             el(2)-inclination(rad)
c             el(3)-argument of perigee(rad)
c             el(4)-semimajor axis(km)
c             el(5)-eccentricity(-)
c             el(6)-mean anomaly(rad)
c             el(7)-true anomaly(rad)
c             el(8)-(not computed)
c             el(9)-eccentric anomaly(rad)
c      output:x-array of satellite coordinates and velocity.
c             u-argument of latitude(rad)
c             n3-orbit number(if used in orbital elements).
c
      implicit real*8(a-h,o-z)
      character*8 sate
      dimension el(9),x(6),par(9)
      common/el/ al(100,8),dal(100,6)
      common/elmn/ ep(3),eg(3),ei(2),ee(2),ea(2),em(4),djne,sate,
     1n1,jp1,jg1,ji1,je1,ja1,jm1
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
      el(1)=elemen(eg,enl,jg1)
      el(2)=elemen(ei,enl,ji1)
      el(3)=elemen(ep,enl,jp1)
      el(4)=elemen(ea,enl,ja1)
      el(5)=elemen(ee,enl,je1)
      if(el(5).lt.0d0) el(5)=0d0
      n3=0
      a=0d0
      am=elemen(em,enl,jm1)
      if(n1.le.0) goto 1
      n2=idint(am)
      if(am.lt.0d0) n2=n2-1
      n3=n1+n2
    1 continue
      am=dmod(am,1d0)*dpi
      if(i.eq.3) a=antk( el(3),el(5))
      el(6)=angle(am-a,dpi)
      if(k.eq.0)goto 2
      el(1)=el(1)+dal(n,2)
      el(2)=el(2)+dal(n,3)
      el(3)=el(3)+dal(n,1)
      el(4)=el(4)+dal(n,6)*1d3
      el(5)=el(5)+dal(n,4)
      el(6)=el(6)+dal(n,5)
    2 call epler(el(6),el(5),el(7),el(9))
      u=angle(el(3)+el(7),dpi)
      call covel2(i,el,x,par)
      return
      end
      subroutine sapla(dmjd,sod)
c
c     evaluation of the mean sidereal time according to newcomb
c     input: dmjd-modified julian date
c     output: sod-sidereal time at the greenwich meridian(rad)
c
      implicit real*8 (a-h,o-z)
      data   dpi,cdr/6.283185307179585d0,1.745329252d-2/
```

```
      data  dhg,hg1/7.292115147d-5,6.300387487d0/
      mjd=idint(dmjd)
      sod=0.2769194d0+0.00273790926d0*( mjd-15019.5d0)
      sod=dmod(sod,1d0)*dpi+hg1*dmod(dmjd,1d0)
      return
      end
c
      subroutine short(dr,di,dl,dm,el)
c
c     computation of the short periodic perturbations by kozai's
c     formula (see sel/i,p.146)
c     input:el-array of elements,computed by function elemen.
c     subroutines : angle
c     output: dr-perturbation of the radius -vector (km)
c             di-perturbation of the inclination(rad)
c             dl-perturbation of the argument of latitude(rad)
c             dm-perturbation of the ascending node(rad)
c
      implicit real*8 (a-h,o-z)
      dimension el(9),aj(21)
      data aj(2),aj(3),aj(4),aj(5)/1.082639d-3,-2.546d-6,-1.649d-6,
     *        -0.210d-6/
      data  rz,dmi/6.37814d3,398601.3d0/
      data twopi,cdr/6.283185307179585d0,1.745329252d-2/
      eo=el(5)
      om=el(3)
      aa=el(4)/rz
      ce=dcos(el(9))
      am=el(6)
      v=el(7)
      sv=dsin(v)
      cv=dcos(v)
      ee=1.-eo*eo
      et=dsqrt(ee)
      si=dsin(el(2))
      co=dcos(el(2))
      s2=si*si
      po=aa*ee
      h2=1.-eo*ce
      g2=1.5*s2-1.
      pj=1.5d0*aj(2)/po
      p2=eo/(1.+et)
      u=2.*om+v
      u2=u+v
      u3=u2+v
      cu2=dcos(u2)
      dr=pj/3.*(g2*(1.-h2/et+p2*cv)+0.5*cu2*s2)
      dr=dr*rz
      pj=pj/po
      di=0.5*si*pj*co*(cu2+eo*(dcos(u)+1./3.*dcos(u3)))
      cu2=dsin(u2)
      su=dsin(u)
      su3=dsin(u3)
      h2=angle(v-am+eo*sv,-twopi)
      dl=pj*(0.5*((7./6.*s2-1.)*cu2+eo*((5./3.*s2-1.)*su+(s2-1.)
     1*su3/3.))-g2/3.*((1.-et)*sv*cv+eo*p2*p2*sv)+(2.5*s2-2.)*h2)
      dm=pj*co*(0.5*(cu2+eo*(su+su3/3.))-h2)
      return
      end
c
      real function vect*8 (x,y,z)
      real*8 x,y,z
c
c     lenght of the vector with
c     components x,y,z
c
      vect=dsqrt(x*x+y*y+z*z)
      return
      end

      double precision function dthg83(dmj)
c
c  pocita pravy greenw. hvezdny cas pro okamzik  dmj
c  entry dthgm - pocita stredni hvezdny cas
c  nutna procedura nuta83 (pocita nutaci)
c
      implicit real*8(d)
      common/nu/dmi,d(4)
      call nuta83(dmj)
      dthg83=dmi
      goto 1
      entry dthgm(dmj)
      dthg83=0.
    1 dvepi=6.2831853071796d0
      dzl=dmod(dmj,1d0)
      dt=(dmj-dzl-51544.5)/36525.
      t=dt
      dzl=(dble((-5.9e-15*t+5.9006e-11)*t)+1.00273790935d0)*dzl*dvepi
      dth=(-6.2e-6*t+9.3104e-2)*t
      dth=dmod(((dth+8640184.812866d0)*dt+24110.54841d0)/
     /13750.987083134d0+dthg83+dzl,dvepi)
      if(dth.lt.0d0) dth=dvepi+dth
      dthg83=dth
      return
      end
      subroutine nuta83(dmj)
c
c  pocita nutaci podle wahra pro modifikovany juliansky okamzik  dmj
c  vystup: dmi=dpsi*cos(deps), dni=dpsi*sin(deps)
c          ddeps, dpsi - nutace ve sklonu a delce
c          deps - okamzity sklon ekliptiky
c
      implicit real*8(d)
      real fdl(5)
      integer arg(106),p(106),e(106),po(15),pt(15),eo(15),et(15),no(15)
      common/nu/dmi,dni,ddeps,dpsi,deps
c  argumenty
      data arg/55556,55557,35756,75355,35757,64545,53736,75356
     1,55737,56555,56737,54737,55736,75535,55735,57555,56556
     2,57737,54556,35576,54736,75536,56736,65545,76535,55376
     3,56375,56557,45566,56735,55757,65555,55756,65757,65535
     4,45757,55575,65556,45556,45777,65756,55777,75555,65737
     5,75757,55755,45756,45576,65536,45776,66535,56757,54757
     6,65777,65575,75737,55576,55776,65736,55536,64555,75756
     7,56535,65355,55565,66555,65755,64757,44777,35556,85757
     8,54777,66757,45736,75556,65557,85555,55767,45557,65515
     9,35777,45797,75515,66737,65776,35797,45957,64535,75736
     a,75777,65576,55937,85737,65735,56756,44576,55356,55747
     b,56575,65335,54756,66536,65375,75575,55797,56565/
c  koeficienty nutace v delce (0.0001")
      data p/0,0,46,11,-3,-3,-2,1,5*0,48,-22,0,-15,0,-12,-6,-5
     1,4,4,-4,1,1,-1,1,1,-1,4*0,-158,123,63,2*0,-59,-51,-38,29
     2,29,-31,26,21,16,-13,-10,-7,7,-7,-8,6,6,-6,-7,6,-5,5,-5,-4
     3,4,-4,-3,3,-3,-3,-2,-3,-3,2,-2,2,-2,2,2,1,-1,1,-2,-1,1,-1,-1
     4,1,1,1,-1,-1,1,1,-1,1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,1/
c  koeficienty nutace ve sklonu (0.0001")
      data e/0,0,-24,0,1,0,1,0,5*0,1,0,0,9,0,6,3,3,-2,-2,7*0,4*0
     1,-1,-53,-2,2*0,26,27,16,-1,-12,13,-1,-10,-8,7,5,0,-3,3,3,0
```

```
     2,-3,3,3,-3,3,0,3,5*0,5*1,-1,1,-1,1,0,-1,-1,0,-1,1,0,-1,1,1
     3,0,0,-1,17*0/
c  indexy a koeficienty casove zavislych clenu (0.0001")
      data no/1,2,9,10,11,12,13,16,18,31,32,33,34,38,39/
      data po/-171996,2062,-13187,1426,-517,217,129,17,-16,-2274
     +,712,-386,-301,63,-58/
      data eo/92025,-895,5736,54,224,-95,-70,0,7,977,-7,200,129,-33,32/
c  casove zmeny (0.00001")
      data pt/-1742,2,-16,-34,12,-5,1,-1,1,-2,1,-4,0,1,-1/
      data et /89,5,-31,-1,-6,3,0,0,0,-5,0,0,-1,0,0/
c
      dvepi=6.283185308d0
      drovt=206264.80625d0
      dt=(dmj-51544.5)/36525.
      t=dt
c  fdl(i) = omega, d, f, l1, l  (v otockach)
      fdl(1)=dmod(((6.2d-9*dt+5.752d-6)*dt-5.3726007245d0)*dt+
     +0.3473458951d0,1d0)*dvepi
      fdl(2)=dmod(((1.5d-8*dt-5.317d-6)*dt+1236.8530874444d0)*dt+
     +0.8273621196d0,1d0)*dvepi
      fdl(3)=dmod(((8.5d-9*dt-1.0229d-5)*dt+1342.2278264946d0)*dt+
     +0.2590886397d0,1d0)*dvepi
      fdl(4)=dmod(((-9.3d-9*dt-4.452d-7)*dt+99.9973620556d0)*dt+
     +0.9931325648d0,1d0)*dvepi
      fdl(5)=dmod(((4.9d-8*dt+2.4159d-5)*dt+1325.5524094390d0)*dt+
     +0.3748971705d0,1d0)*dvepi
c  vypocet koeficientu casove zavislych clenu
      do 1 j=1,15
      i=no(j)
      p(i)=po(j)+ifix(t*pt(j)/10.)
    1 e(i)=eo(j)+ifix(t*et(j)/10.)
c  nutace
      ddeps=0.
      dpsi=0.
      do 2 n=1,106
      alf=0.
      i=arg(n)
      do 3 j=1,5
      alf=alf+fdl(j)*(mod(i,10)-5)
    3 i=i/10
      ddeps=ddeps+dfloat(e(n))*cos(alf)
    2 dpsi=dpsi+dfloat(p(n))*sin(alf)
      ddeps=ddeps/drovt/1d4
      dpsi=  dpsi/drovt/1d4
      deps=((dble((1.813e-3*t-5.9e-4)*t)-46.815d0)*dt+84381.448d0)/drovt
     ++ddeps
      dmi=dpsi*dcos(deps)
      dni=dpsi*dsin(deps)
      return
      end
```