1 Introduction

In this document the arrival-time analysis is discussed in case we are dealing with a pulsar in a binary system. The induced accelerations along the orbit introduce additional corrections to be applied to the photon arrival time at solar system barycentre. The method has been implemented a long time ago in the PUL software under algorithm identifier PUL-AL-020, so this report merely gives some additional information. A minor change with respect to the former PUL-AL-020 had been made : the iteration process, in which the eccentric anomaly equation is solved, is accelerated. The fundamental theory can be found in Blandford & Teukolsky.

2 Definitions

E	Eccentric anomaly
e	Eccentricity of orbit
i	Inclination of orbit
a_1	Semimajor axis of pulsar orbit (around mass center) in lightseconds
P_{orb}	Orbital period
Φ_{orb}	Orbital phase
t	Barycentric arrival-time (TDB scale) of photon
t_0	Barycentric time (TDB scale) of periastron
t_e	Nominal epoch of pulsar ephemeris (TDB scale)
ω_0	Longitude of periastron at t_0
$\dot{\omega}_0$	Advance of periastron at t_0
γ	Time-dilation and gravitational redshift parameter
ν	Frequency of pulsar
$\dot{\nu}$	First time derivative of pulsar frequency
$\ddot{\nu}$	Second time derivative of pulsar frequency

3 Literature


```
[1] ApJ 205:580-591,1976 Arrival-time analysis for a pulsar in a binary system
                         R.Blandford & S.Teukolsky
[2] ApJ 206:L53-L58,1976 Further observations of the binary pulsar PSR 1913+16
                         J.H.Taylor et al.
[3] PUL-AL-020           Correction to binary focus
                         M.Busetta
```


4 Description of the method

First we have to solve the eccentric anomaly E from the following transcedental equation :

$$E - e \cdot \sin(E) - \Phi_{orb} = 0 \tag{1}$$

The orbital phase is given by :

$$\Phi_{orb} = 2\pi \cdot (t - t_0)/P_{orb} \tag{2}$$

Equation (1) is solved iteratively using a Newton-Raphson iteration technique. The longitude of periastron at time t (in TDB) is given by :

$$\omega = \omega_0 + \dot{\omega}_0 \cdot (t - t_0) \tag{3}$$

The following quantities can be defined :

$$\alpha = a_1 \cdot \sin(i) \cdot \sin(\omega) \tag{4}$$
$$\beta = \sqrt{1-e^2} \cdot a_1 \cdot \sin(i) \cdot \cos(\omega) \tag{5}$$

Define also :

$$Q = \alpha \cdot \big(\cos(E) - e\big) + (\beta + \gamma) \cdot \sin(E) \tag{6}$$
$$R = -\alpha \cdot \sin(E) + \beta \cdot \cos(E) \tag{7}$$
$$S = 1/\big(1 - e \cdot \cos(E)\big) \tag{8}$$

Then the total delay t_d due to the pulsar acceleration in its orbit around the mass center can be expressed as :

$$t_d = Q - R \cdot Q \cdot S \cdot 2\pi/P_{orb} \tag{9}$$

See for example eq. (2.33) of [1]. The time $t_{cor} = t - t_d$ is subsequently used in the calculation of the pulsar phase according to :

$$\Phi = \text{mod}(\nu \cdot \Delta t + 1/2 \cdot \dot{\nu} \cdot \Delta t^2 + 1/6 \cdot \ddot{\nu} \cdot \Delta t^3, 1.) \tag{10}$$

in which $\Delta t = t_{cor} - t_e$.

A Technical aspects

To solve the transcedental equation for the eccentric anomaly we have to use iterative methods. Used here is a Newton-Raphson method. A good starting point for E can be found if we write E as a perturbation series in e ie. $E = \sum_{n=0}^{\infty} e^n \cdot E_n$ (11). Solving equation (1) with (11) substituted gives up to and including second order $(n = 2)$ the following approximation to the real value of E.

$$E^0 = E^{app} = \Phi_{orb} + e \cdot \sin(\Phi_{orb}) + e^2 \cdot \sin(\Phi_{orb}) \cdot \cos(\Phi_{orb}) \tag{12}$$

The following Newton-Raphson scheme is used for solving E :

$$E^{n+1} = E^n - \frac{f(E^n|e, \Phi_{orb})}{df/dE(E^n|e, \Phi_{orb})} \tag{13}$$

in which

$$f(E|e, \Phi_{orb}) = E - e \cdot \sin(E) - \Phi_{orb} \tag{14}$$
$$df/dE(E|e, \Phi_{orb}) = 1 - e \cdot \cos(E) \tag{15}$$

If the absolute value of the difference of the n^{th} and $n+1^{th}$ approximation to E is less than 10^{-12} the iteration process terminates. Below follow the FORTRAN and C source codes of the software used by the COMPTEL - and EGRET-team and in the TEMPO software package (from Princeton) in which the time delay t_d is calculated. All these implementations are equivalent. However,all have in common that the iteration process is not efficiently coded in the sense that the $df/dE(E|e,\Phi_{orb})$ in (13) is not updated each successive iteration step and remained at its initial value (however, the final value of E is not different in both cases). Tests indicate that the iteration process can be accelerated more than a factor of 2 (in case of e close to 1), if also the denominator in (13) is adapted each iteration step. I have implemented this in my local test- environment.

A.1 COMPTEL source code : PUL-Al-020

```
Compass source library.file-id: 'PULALG01.PULA20'
C**********************************************************************
C* Institute:          SSD        *             *   Subroutine      *
C*                               *             *                   *
C* PPPPPPP    UUU  UUU   LLL      *     GRO     *   PULA20          *
C* PPPPPPPP   UUU  UUU   LLL      *   COMPTEL   *                   *
C* PPP  PPP   UUU  UUU   LLL      *             *   Revision:       *
C* PPP  PPP   UUU  UUU   LLL      *             *                   *
C* PPPPPPPP   UUU  UUU   LLL      ***************   1.0             *
C* PPPPPPP    UUU  UUU   LLL      *             *                   *
C* PPP        UUU  UUU   LLL      *             *   Author:         *
C* PPP        UUU  UUU   LLL      *   COMPASS   *                   *
C* PPP        UUUUUUUU   LLLLLLLL *             *   C. JENKINS      *
C* PPP         UUUUUU    LLLLLLLL *             *                   *
C*                               *             *   Date:           *
C* Pulsar Analysis               *             *                   *
C*                               *             *   9-JUL-91        *
C*                               *             *                   *
C**********************************************************************
C* Subroutine History:                                                *
C*                                                                    *
C* date           programmer     reason for change                    *
C* 28/11/89       CJ             Initial Version                      *
C*                                                                    *
C*                                                                    *
C* Calls: None                                                        *
C*                                                                    *
C* Inputs:                                                            *
C*                                                                    *
C* INTEGER LUNLOG         Logical unit for readable messages          *
C* INTEGER SSBTIM(2)      Time at SSB from PUL-AL-004                 *
C* DOUBLE  BIPE(2)        Binary orbital period + error               *
C* DOUBLE  BIPED(2)       Binary orbital period derivative + error    *
C* DOUBLE  BIOE(2)        Binary orbit eccentricity + error           *
C* DOUBLE  SEMA(2)        Projected semi-axis major + error           *
C* DOUBLE  PERI(2)        Argument of the periastron + error          *
C* DOUBLE  PERID(2)       Advance of the periastron + error           *
C* INTEGER TPERI(2,2)     Time of the periastron + error              *
C* DOUBLE  G              Measure of relativistic time dilation       *
```

```
C*                                                                  *
C* Outputs:                                                         *
C*                                                                  *
C* INTEGER CTIME(2)        The corrected time to focus              *
C* INTEGER IERR            Error return number                      *
C*                                                                  *
C* Description:                                                     *
C*                                                                  *
C* This routine implements the algorithm PUL-AL-020 to calculate    *
C* the correction of the SSB times to the focus of the orbit in the *
C* case of a binary pulsar.                                         *
C*                                                                  *
C*******************************************************************
C
      SUBROUTINE PULA20 ( LUNLOG, SSBTIM, BIPE, BIPED, BIOE, SEMA,
     1                    PERI, PERID, TPERI, G,  CTIME, IERR )
C
C Declaration of variables in argument list
C
      INTEGER          LUNLOG, SSBTIM(2), TPERI(2,2), CTIME(2), IERR
      DOUBLE PRECISION BIPE(2), BIPED(2), BIOE(2), SEMA(2), PERI(2)
      DOUBLE PRECISION PERID(2), G
C
C Local
C
      DOUBLE PRECISION TWOPI, DAYSEC, T, TT, PHASE, EP
      DOUBLE PRECISION DENOM, DEP, OMEGA, SOM, COM
      DOUBLE PRECISION ALPHA, BETA, SBE, CBE, Q
      DOUBLE PRECISION DELAY, PHORB
C
C-------------------------------------------------------------------
C
C Set error return to 0
C
      IERR = 0
C
C Set constant parameters
C
      TWOPI = 6.28318530717958648D0
      DAYSEC = 86400.D0
C
C Compute orbit phase at the input time
C
      T = ( SSBTIM(1) + (DFLOAT(SSBTIM(2))/DAYSEC/8000.D0) ) -
     &    (TPERI(1,1) + (DFLOAT(TPERI(2,1))/DAYSEC/8000.D0))
      TT    = T * DAYSEC / BIPE(1)
      PHASE = TWOPI * DMOD (TT - 0.5D0*BIPED(1)*TT*TT, 1.D0)
```

```
C
C Estimate the eccentric anomaly by iteration
C
      EP    = PHASE + BIOE(1)*DSIN(PHASE)*(1.D0+BIOE(1)*DCOS(PHASE))
      DENOM = 1.D0 - BIOE(1)*DCOS(EP)
C
      DEP = 1.D0
 100  CONTINUE
      IF ( DABS(DEP) .LE. 1.D-12 ) GO TO 200
         DEP = (PHASE - (EP-BIOE(1)*DSIN(EP)))/DENOM
         EP = EP + DEP
         GO TO 100
 200  CONTINUE
C
C Determination of the longitude of periastron at the input time
C
      OMEGA = PERI(1) + PERID(1)*T
C
C Evaluation of the delay in pulse arrival time due to orbital effects
C and the orbital phase corresponding to the arrival time
C
      SOM  = DSIN(OMEGA)
      COM  = DCOS(OMEGA)
      ALPHA = SEMA(1)*SOM
      BETA = SEMA(1)*COM*DSQRT(1.D0-BIOE(1)*BIOE(1))
      SBE  = DSIN(EP)
      CBE  = DCOS(EP)
      Q    = ALPHA * (CBE-BIOE(1)) + (BETA+G)*SBE
C
      DELAY = -Q+(TWOPI/BIPE(1))*Q*(BETA*CBE-
     &        ALPHA*SBE)/(1.D0-BIOE(1)*CBE)
      PHORB = DMOD(2.D0 + (PHASE + OMEGA)/TWOPI, 1.D0)
C
C Correction of the arrival time for the orbital effect
C
      CTIME(1) = SSBTIM(1)
      CTIME(2) = SSBTIM(2) + DELAY*8000.D0
C
      IF (CTIME(2) .GE. 691200000) THEN
          CTIME(2) = CTIME(2) - 691200000
          CTIME(1) = CTIME(1) + 1
      ELSE IF (CTIME(2) .LT. 0) THEN
          CTIME(2) = CTIME(2) + 691200000
          CTIME(1) = CTIME(1) - 1
      END IF
C
      RETURN
```

```
      END
```

A.2 EGRET source code provided by J.Fierro

```
/**********************************  BNRYBT  **********************************
*  Compute orbit phase at the input time (relative to periastron) and
*  estimate the eccentric anomaly.
*  Compute delay in pulse arrival time due to orbital effects and orbital
*  phase in turns relative to ascending node.
*  (See R. Blandford and S. Teukolsky, Ap.J. 205, 580 (1976) and
*       J.H. Taylor et al., Ap.J. Lett. 206, L53 (1976).)
******************************************************************************/
double
bnrybt(tdb)
double tdb;
{
double t, tpb, phase, ep, denom, dep, omega, sbe, cbe, q;
double alpha, beta;
t = tdb - t0;
tpb = t*SECDAY/pb;
phase = fmod(tpb - 0.5*pbdot*tpb*tpb, 1.0)*twopi;
if (phase < 0)
phase = phase + twopi;
ep = phase + e*sin(phase)*(1 + e*cos(phase));
denom = 1 - e*cos(ep);
/* Iterate to calculate EP  (Pat Wallace's method?) */
dep = 1;
while(fabs(dep) > 1e-12)
dep = (phase - (ep-e*sin(ep)))/denom;
ep = ep + dep;
omega = omz + omdot*t;
alpha = a1*sin(omega);
beta = a1*cos(omega)*sqrt(1 - e*e);
sbe = sin(ep);
cbe = cos(ep);
q = alpha*(cbe-e) + (beta+gamm)*sbe;
torb = -q + (twopi/pb)*q*(beta*cbe - alpha*sbe)/(1 - e*cbe);
return (fmod(2 + (phase + omega)/twopi, 1.0));
}
```

A.3 TEMPO subroutine "bnrybt.f"

```
      subroutine bnrybt(torb,fctn)

C  This version uses the method of Blandford and Teukolsky (ApJ 205,
C  580,1976)--including a solution for gamma.  For notation, see Taylor
C     et al, ApJ Lett 206, L53, 1976.

      implicit real*8 (a-h,o-z)
      parameter (twopi=6.28318530717958648d0)
      parameter (rad=360.d0/twopi)
      include 'dim.h'
      real*8 fctn(npap1),k
      include 'dp.h'
      include 'orbit.h'
 C  Allow for more than one orbit:
      torb=0.
      jbin=1
      if(nbin.ge.9) jbin=nbin-7
      do 20 i=1,jbin
      tt0=(ct-873.5d0-t0(i))*86400.d0
      fac=1.d0
      if(i.eq.2) fac=-1.6072
      orbits=tt0/pb(i) - 0.5d0*(fac*pbdot+xpbdot)*(tt0/pb(i))**2
      norbits=orbits
      if(orbits.lt.0.d0) norbits=norbits-1
      phase=twopi*(orbits-norbits)
      omega=(omz(i)+omdot*tt0/(86400.d0*365.25d0))/rad
C  Use Pat Wallace's method of solving Kepler's equation
      ep=phase + e(i)*sin(phase)*(1.d0+e(i)*cos(phase))
      denom=1.d0-e(i)*cos(ep)
10    dep=(phase-(ep-e(i)*sin(ep)))/denom
      ep=ep+dep
      if(abs(dep).gt.1.d-12) go to 10
      bige=ep
      if(nbin.eq.6) then
        u=ep
        su=sin(u)
        cu=cos(u)
        onemecu=1.d0-e(i)*cu
        cae=(cu-e(i))/onemecu
        sae=sqrt(1.d0-e(i)**2)*su/onemecu
        ae=atan2(sae,cae)
        if(ae.lt.0.0) ae=ae+twopi
        ae=twopi*orbits + ae - phase
        an=twopi/pb(i)
        k=omdot/(rad*365.25d0*86400.d0*an)
```

```
          omega=omz(i)/rad + k*ae
        endif
        tt=1.d0-e(i)*e(i)
        som=sin(omega)
        com=cos(omega)
        alpha=a1(i)*som
        beta=a1(i)*com*sqrt(tt)
        sbe=sin(bige)
        cbe=cos(bige)
        q=alpha*(cbe-e(i)) + (beta+gamma)*sbe
        r=-alpha*sbe + beta*cbe
        s=1.d0/(1.d0-e(i)*cbe)
        torb=-q+(twopi/pb(i))*q*r*s + torb
        j=0
        if(i.eq.2) j=17
        if(i.eq.3) j=17+5
        fctn(9+j)=f0*(som*(cbe-e(i)) + com*sqrt(tt)*sbe)
        fctn(10+j)=-f0*(alpha*(1.+sbe*sbe-e(i)*cbe)*tt -
     +    beta*(cbe-e(i))*sbe)*s/tt
        fctn(11+j)=-f0*(twopi/pb(i))*r*s*86400.d0
        fctn(12+j)=fctn(11+j)*tt0/(86400.d0*pb(i))
        fctn(13+j)=f0*a1(i)*(com*(cbe-e(i)) - som*sqrt(tt)*sbe)
        if(i.eq.1) then
          fctn(14)=fctn(13)*tt0/(rad*365.25d0*86400.d0)
          fctn(15)=f0*sbe
          fctn(18)=0.5d-6*fctn(12)*tt0
          fctn(24)=tt0*fctn(9)
          fctn(25)=tt0*fctn(10)
        endif
        if(i.ge.2) fctn(18)=fctn(18) + fac*0.5d-6*fctn(12+j)*tt0
20      continue

        return
        end
```

Document Title	**Timing analysis in case of a binary pulsar**
Document ID	**COM-RP-ROL-DRG-66**
Issue No	**1**
Issue Date	**25 November 1994**
Prepared by	**Drs. L.M.Kuiper**
Approved by	**Dr. W. Hermsen**
Authorized by	

Contents