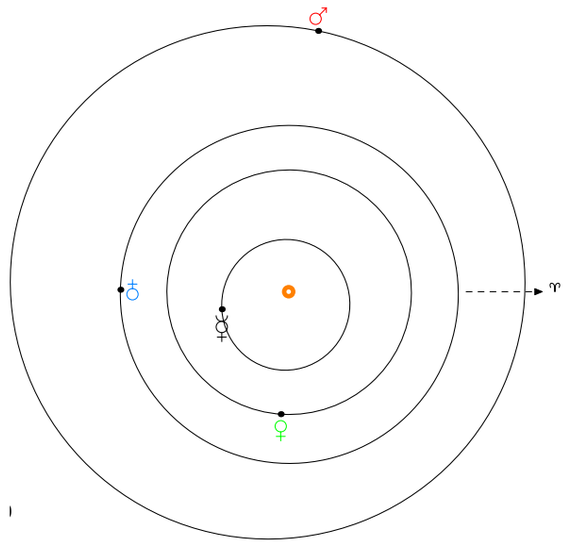


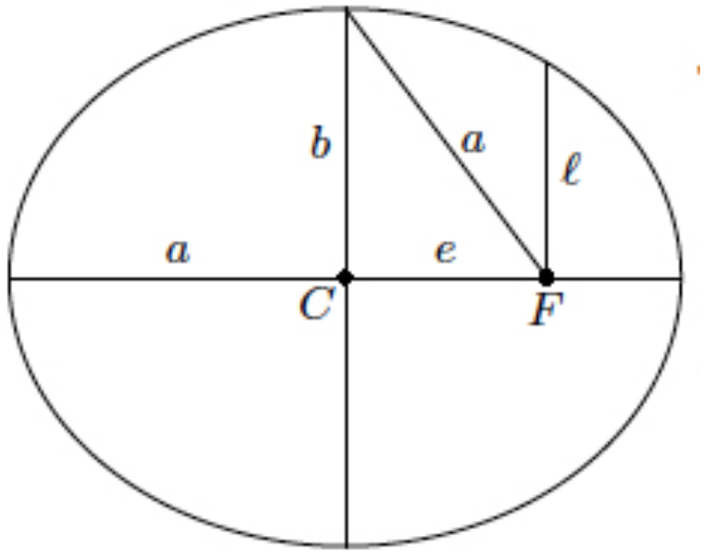
Planety

Prawa Keplera

Elipsa

- Merkury, Wenus, Ziemia, Mars,…
- Bez inklinacji
- Orbity, niezmienniki
  - Mimośród
  - Semilatus rektum: ℓ

    semimajor axis: a



$$a = \frac{l}{1-\varepsilon^2}\,, \qquad b = \frac{l}{\sqrt{1-\varepsilon^2}}$$

$$e = \varepsilon \cdot a\,, \quad a^2 = b^2 + e^2$$

Elipsa

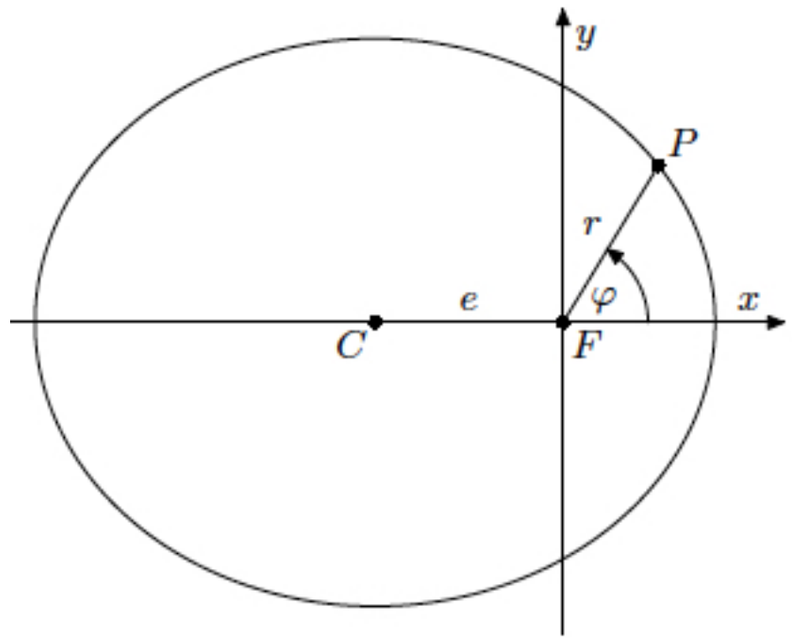
$$\frac{x^2}{a^2} + \frac{y^2}{a^2} = 1$$

$$x^2 + (\frac{ay}{b})^2 = a^2$$

„okrąg skalowany y-czynnikiem”

$$b/a = \sqrt{1 - \varepsilon^2}$$

Równanie elipsy dla F w centrum układu współrzędnych:



$$r = \frac{l}{1 + \varepsilon \cos \phi}$$

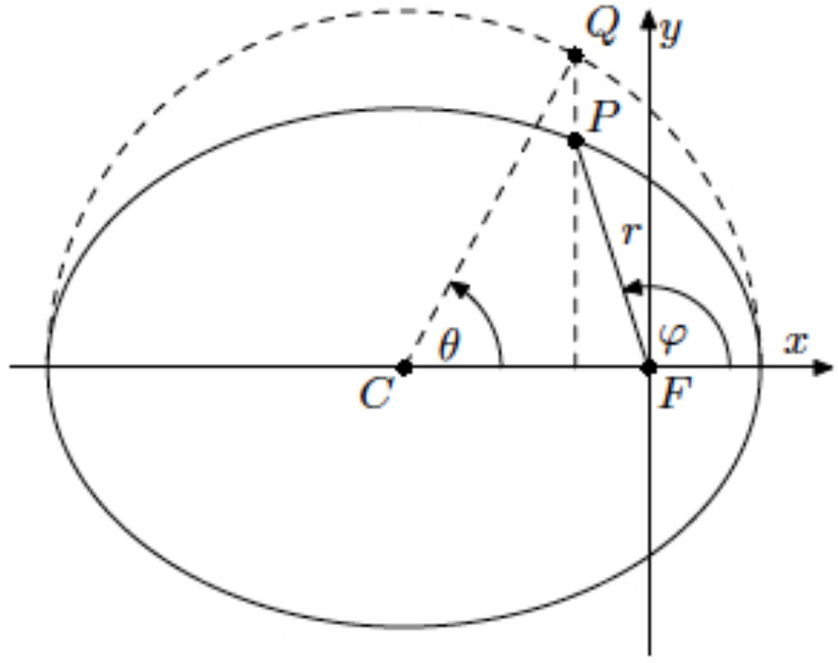
Elipsa

$$\sin\theta = \frac{\sqrt{1-\varepsilon^2}}{1+\varepsilon\cos\phi}$$

$$\cos\theta = \frac{\cos\phi+\varepsilon}{1+\varepsilon\cos\phi}$$

$$\sin\phi = \frac{\sqrt{1-\varepsilon^2}\sin\theta}{1-\varepsilon\cos\theta}$$

$$\cos\phi = \frac{\cos\theta-\varepsilon}{1-\varepsilon\cos\theta}$$



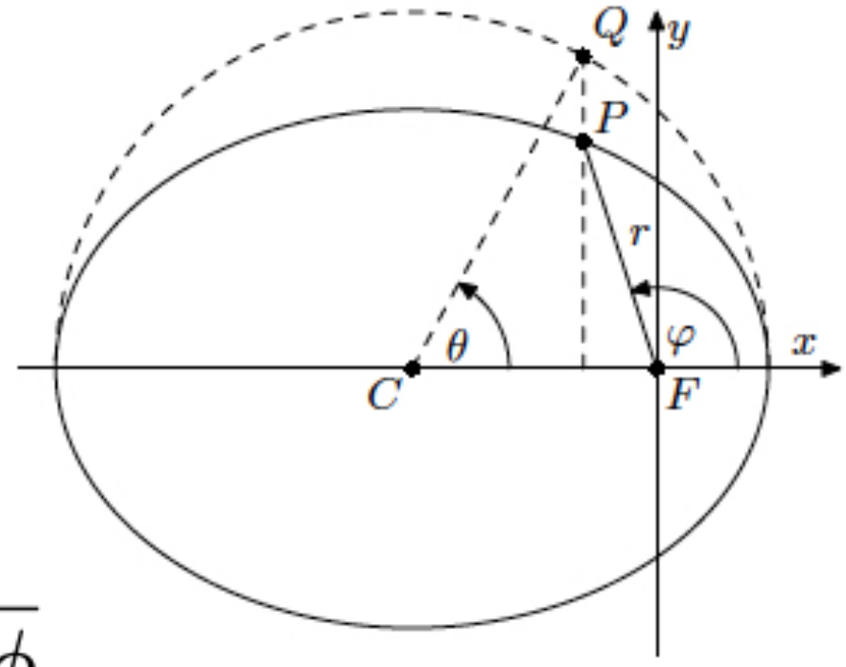
$$\tan\frac{\phi}{2} = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}\tan\frac{\theta}{2}$$

Jak?

Punkt Q:

$$a\sin\theta = ay/b$$

$$a\cos\theta = x + e$$



Więc

$$\sin\theta = y/b = \frac{r\sin\theta}{b} = \frac{\sqrt{1-\varepsilon^2}}{1+\varepsilon\cos\phi}$$

$$\cos\theta = \frac{x+e}{a} = \ldots$$

Perihelium, aphelium

$$r_{min} = \frac{l}{1+\varepsilon}$$

$$r_{max} = \frac{l}{1-\varepsilon}$$

R. Keplera

$$\boldsymbol{L} = \boldsymbol{r} \times \dot{\boldsymbol{r}}$$

$$\dot{\boldsymbol{L}} = \boldsymbol{r} \times \ddot{\boldsymbol{r}} = \boldsymbol{0}$$

$$\sqrt{l\alpha}\, dt = r^2\, dt \qquad \alpha = G(M+m) \tag{II}$$

$$\theta - \varepsilon \sin\theta = \frac{2\pi}{T}(t - t_0)$$

(Równanie Keplera)
Zadanie

$$\frac{a^3}{T^2} = \frac{G(M+m)}{4\pi^2} \tag{III}$$

JD2000

Dzień Juliański: 1-1-4712 —> ...

jd = 2 456 579 = 20.03.2019 - 1szy dzień wiosny

dzień tygodnia = 1 + JD mod 7

Program

```
program inner_planets
  implicit none
  real, parameter           :: PI  = 3.14159265358979324
  real, parameter           :: DEG = (180.0/PI)
  real, parameter           :: RAD = PI/180.0
  character*7, dimension(4) :: planet
  real, dimension(4)        :: e, de, L, dL, w, dw
  !
  integer                   :: i, x, year, month, day, j, julian
  real                      :: T, g2j, y, m, d
  real, dimension(4)        :: ec, ma, ml, th, v
  ! ------------------------------------------------------------------------
  ! set constants - Keplerian elements from Standish + corrections
  ! (For other planets see the end of the file)
  planet = ["Mercury",  "Venus  ",   "Earth  ", "Mars   "]
  e      = [0.20563661, 0.00676399, 0.01673163, 0.09336511]
  de     = [0.00002123,-0.00005107,-0.00003661, 0.00009149]
  L      = [   252.25166724,   181.97970850,   100.46691572,    -4.56813164]
  dL     = [149472.67486623, 58517.81560260, 35999.37306329, 19140.29934243]
  w      = [77.45771895, 131.76755713, 102.93005885, -23.91744784]
  dw     = [ 0.15940013,   0.05679648,   0.31795260,   0.45223625]
  ! ------------------------------------------------------------------------
  print*, "solar"
  print*, "True anomaly of Mercury, Venus, Earth and Mars"
  do
     print*, "Input date: day month year, e.g., 20 03 2019"
     read(*,*,err=1) day, month, year
     if (year<-3000 .or. year>3000) then
        print*, "Error: year outside allowed range -3000 <= year <= +3000!"
        stop
     end if
     write(*,"(i4,1h-,i2,1h-,i2)") year, month, day
     go to 2
  end do
1 print*,">>> Wrong input date. Try again"
2 continue
```

```
if (year==1582 .and. month==10 .and. day>4 .and. day<15) then
     print*, "This date never existed!"
     stop
  end if
  !
  ! julian day
  j = Julian(year,month,day)

  ! julian century
  x = j - Julian(2000,1,1) !2451545 = days since 2000-01-01
  T = (x+0.)/36525.0                ! julian centuries

  do i=1,4
     ec(i)  = e(i)+de(i)*T          ! eccentricity
     ml(i) = L(i)+dL(i)*T           ! mean longitude
     v(i)  = w(i)+dw(i)*T           ! longitude of perihelion
     ma(i)  = (ml(i)-v(i))*RAD      ! mean anomaly
     th(i) = theta(ma(i),ec(i))     ! eccentric anomaly
     ! true anomaly
     ma(i)  = 2*atan(sqrt((1+ec(i))/(1-ec(i)))*tan(th(i)/2))*DEG+v(i)
  end do

print*, "julian day: ", j
  do i=1,4
     write(*,"(a, a7,1x,2f9.3)") "Perihelium: ", planet(i), ma(i), v(i)
  end dD
containS
 …
end program inner_planets
```

```
! using g2j(...)
  !y = year; m = month; d = day + 0.5  ! add half a day here
  !j = int( g2j(y,m,d) )
  !! Check
!! 1957 October 4.81 should = 2436116.31
!print*, '1957 October 4.81 should = 2436116.31 ',g2j(1957,10,4.81)
!! 333 January 27 at 1200 hrs UT = 1842713.0
!print*, '333 January 1200 hrs UT = 1842713.0 ', g2j(333,1,27.5)
```

```
! ------------------------------------------------------------------
  ! Solve Kepler equation
  ! ------------------------------------------------------------------
  function theta(m, e)           ! solve Kepler equation
    implicit none
    ! theta - eps*sin theta = m
    real theta, m, e, t, d, dt
    if (m >  PI) m = m - 2*PI     ! normalize m to the range -PI <= m <= PI
    if (m < -PI) m = m + 2*PI
    t = m + e*sin(m)
    do while (abs(dt) > 1e-6)
       d  = m - (t - e*sin(t))
       dt = d/(1-e*cos(t))
       t  = t + dt
    end do
    theta = t
    return
  end function theta
```

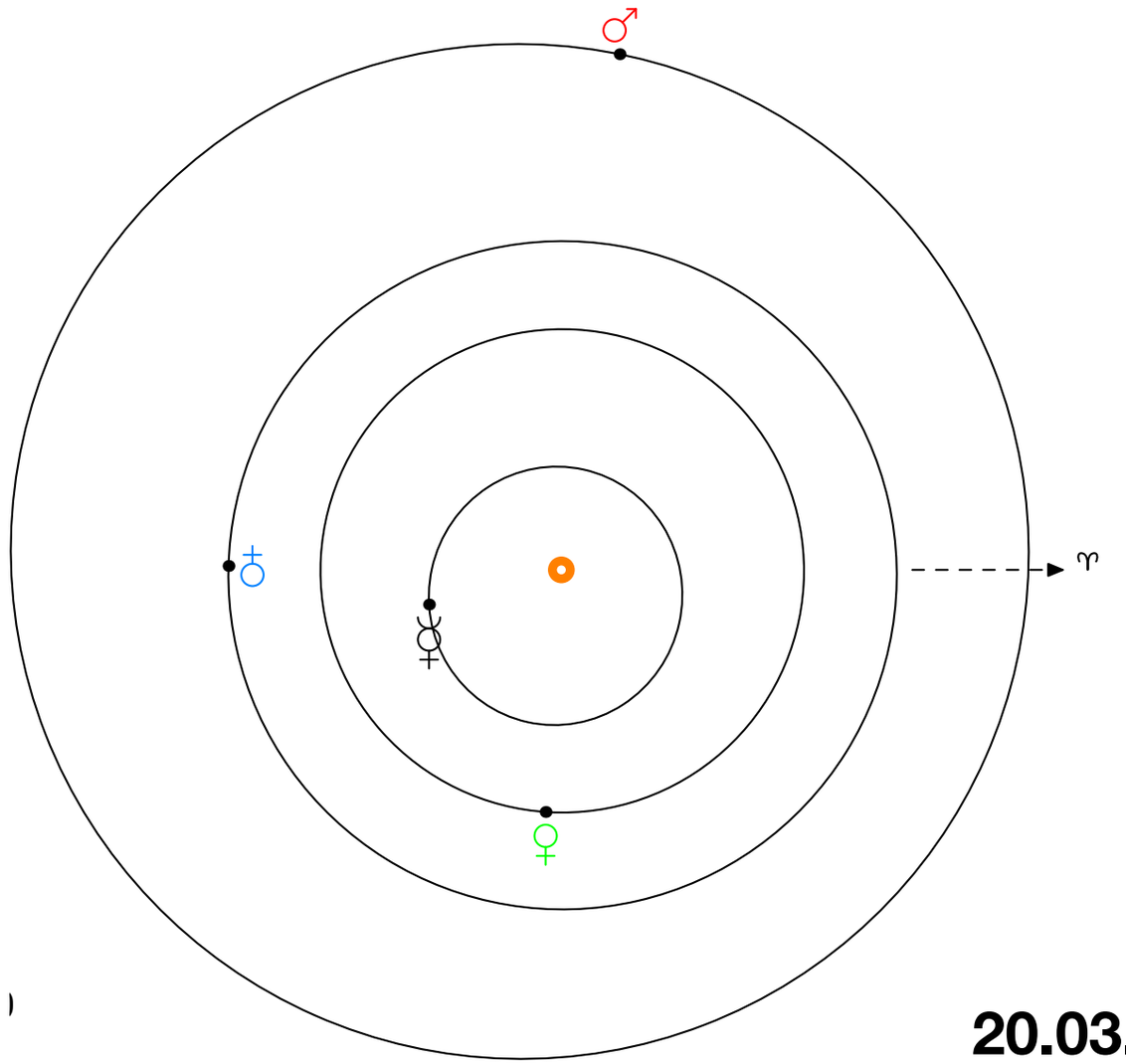
```
! ------------------------------------------------------------------
! Julian - Gregorian date to Julian
! ------------------------------------------------------------------
function Julian(year, month, day)
  implicit none
  integer :: Julian, year, month, day, x
  integer, dimension(0:12)  :: em
  em = [0, 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334]
  !
  ! Julian era
  if (year < 1582 .or. year == 1582 .and. month*100+day < 1005) then
     julian = 1721423 + day + em(month) + (year-1)*365 + gauss_floor(year-1,4)
  else ! Gregorian era
     x = year - 2001
     julian = 2451910 + day + em(month) + x*365 + gauss_floor(x,4) - &
          gauss_floor(x,100) + gauss_floor(x,400)
  end if
  if (month > 2) julian = julian + leap(year) ! correction for leap year
  !
contains
  function gauss_floor(a, b)
    implicit none
    integer a, b, gauss_floor
    ! (a/b) = Gauss parenthesis
    gauss_floor = a/b
    if (a < 0 .and. mod(a,b) .ne. 0) then    ! negative numerator? -
       gauss_floor = gauss_floor - 1         !          substract 1
    end if
  end function gauss_floor
  function leap(y)
    implicit none
    integer y, leap, i
    ! leap = 1 if year is a leap year otherwise leap = 0
    i = 0
    if (mod(y,4) == 0) i = 1
    if (y > 1582 .and. mod(y,100) == 0 .and. mod(y,400) /= 0) i = 0
    leap = i
  end function leap
end function Julian
```

```
! ------------------------------------------------------------------------
! g2j - Gregorian date to Julian
! ------------------------------------------------------------------------
function g2j(year, month, day)
  implicit none
  real, intent(in) :: year, month, day
  real             :: y,m,d,a,b
  real             :: g2j
  !
  ! Here is a Fortran piece of code derived from Jean Meuus' book
  ! "Astronomical Algorithms" to compute the Julian Day (JD) for a
  ! given Gregorian date and time. This formula uses the astronomical
  ! definition of JD in that it includes the year '0', whereas the
  ! historical definition does not.
  !
  ! Convert a Gregorian Date to a Julian Day using Jean Meeus formula 7.1
  !
  y = year;  m = month;  d = day
  if (m < 3) then
     y = y-1
     m = m+12
  end if
  if (y >= 1582) then
     a = int(y / 100)
     b = 2. - a + int(a / 4)
  else
     b = 0
  end if
  g2j = int(365.25 * (y + 4716)) + &
       int(30.6001 * (m + 1)) + d + b - 1524.5
  return

  !! Examples:
  !! 1957 October 4.81 should = 2436116.31
  !! 333 January 27 at 1200 hrs UT = 1842713.0

end function g2j
```

Pozycje planet



20.03.2019 - wiosna

R. Keplera

$$\begin{aligned}
\sqrt{l\alpha}\,dt &= r^2\,d\phi \\
r &= a(1-\varepsilon\cos\theta) \\
&= \frac{l}{1+\varepsilon\cos\phi} \\
\sqrt{l\alpha}\,dt &= a^2(1-\varepsilon\cos\theta)^2\,d\phi \\
&= a^2\sqrt{1-\varepsilon^2}(1-\varepsilon\cos\theta)\,d\theta
\end{aligned}$$

Całkujemy

$$\frac{\sqrt{l\alpha}}{a^2\sqrt{1-\varepsilon^2}}t = \theta - \varepsilon\sin\theta$$

Po jednym obiegu: $\theta = 2\pi \quad \text{oraz} \quad t = T$ **czyli**

$$\frac{2\pi}{T}t = \theta - \varepsilon\sin\theta$$

```
      FUNCTION JD(L,M,N,J1G0)

C   OBLICZA DZIEN JULIANSKI Z DATY KALENDARZA GREGORIANSKIEGO [J1G0
C  = 0] LUB JULIANSKIEGO [J1G0 = 1]  (L = ROK, M = MIESIAC,     N =
C  DZIEN MIESIACA). ALGORYTM DZIALA POPRAWNIE OD 1 MARCA -100100 R.
C  (OBU KALENDARZY).
C
      JD=(L+(M-8)/6+100100)*1461/4+(153*MOD(M+9,12)+2)/5+N-34840408
      IF(J1G0.LE.0)          JD = JD-(L+100100+(M-8)/6)/100*3/4+752
      RETURN
      END
      SUBROUTINE DATA(JD,L,M,N,J1G0)

C   OBLICZA DATE GREGORIANSKA LUB JULIANSKA  (L = ROK, M = MIESIAC
C  [STYCZEN: M=1], N = DZIEN MIESIACA)  Z DNIA JULIANSKIEGO (JD) W
C  POLUDNIE.  STOSOWALNOSC OD JD = -34839655,  TJ. OD ROKU -100100
C  KALENDARZA GREGORIANSKIEGO [J1G0=0] ALBO JULIANSKIEGO [J1G0=1].
C
```

```
J = 4*JD+139361631
IF(J1G0.LE.0)       J = J+(4*JD+183187720)/146097*3/4*4-3908
I = MOD(J,1461)/4*5+308
N = MOD(I,153)/5+1
M = MOD(I/153,12)+1
L = J/1461-100100+(8-M)/6
RETURN
END
```

https://www.astro.uni.torun.pl/~kb/Artykuly/PA/Juldat.htm

Postępy Astronomii
Tom **XXXV** (1987), Zeszyt 4, 275–279.
K. Borkowski. DNI JULIAŃSKIE I DATY KALENDARZOWE