Lunar Phase Computation

by Stephen R. Schmitt

Introduction

During a lunar month (about 29.5) days, the Moon's appearance changes through eight well-known phases that comprise a *lunation*. These phases of the Moon are:

1. New Moon
2. Waxing Crescent
3. First Quarter
4. Waxing Gibbous
5. Full Moon
6. Waning Gibbous
7. Last Quarter
8. Waning Crescent

New Moon, First Quarter, Full Moon, and Last Quarter are the primary phases. The crescent and gibbous phases are intermediate phases. First and Last Quarters occur when the Sun and Moon are 90° degrees apart. The First Quarter and Last Quarter phases are named this way because they occur when the Moon is at one- and three-quarters of a complete cycle. The phases New Moon, First Quarter, Full Moon, and Last Quarter occur when the *ecliptic longitude* of the Moon differs from that of the Sun by 0°, 90°, 180°, and 270°.

The time in days counted from the time of New Moon is called the *Moon's age*.

The *ecliptic longitude* is measured from the vernal equinox along the ecliptic in the direction of the Sun's apparent motion through the stars.

The *ecliptic latitude* is positive north of the ecliptic and negative if south.

Algorithm

This program helps anyone who needs to know the Moon's phase, age, distance, and position along the ecliptic on any date within several thousand years in the past or future. The age of the moon in days as well as its visual phase are given. The Moon's *ecliptic longitude* is calculated as well as the corresponding zodiac constellation.

The Moon's calculated position is based on the Julian Day number corresponding to the calendar date. The date is checked for valid day of the month.

Zeno source code

Zeno 1.2 is an interpreter for the Zeno programming language. It is an easy to learn and is suitable for educational purposes.

- Download Page

```
const PI : real := 3.14159265358979323846

program

    var year, month, day : int
    var tm : real := localtime

    year  := dateyear( tm )
    month := datemonth( tm )
    day   := dateday( tm )

    put "Moon on ", month, '/', day, '/', year
    moon_posit( year, month, day )

end program

%  compute moon position and phase
procedure moon_posit( Y, M, D : int )

    var AG : real                           % Moon's age
    var DI : real                           % Moon's distance in earth radii
    var LA : real                           % Moon's ecliptic latitude
    var LO : real                           % Moon's ecliptic longitude
    var Phase : string
    var Zodiac : string

    var YY, MM, K1, K2, K3, JD : int
    var IP, DP, NP, RP : real

    if not isdayofmonth( Y, M, D ) then
        put "invalid date"
        return
    end if

    % calculate the Julian date at 12h UT
    YY := Y - floor( ( 12 - M ) / 10 )
    MM := M + 9
    if (MM >= 12) then
        MM := MM - 12
    end if

    K1 := floor( 365.25 * ( YY + 4712 ) )
    K2 := floor( 30.6 * MM + 0.5 )
    K3 := floor( floor( ( YY / 100 ) + 49 ) * 0.75 ) - 38

    JD := K1 + K2 + D + 59                  % for dates in Julian calendar
    if (JD > 2299160) then
        JD := JD - K3                       % for Gregorian calendar
    end if

    % calculate moon's age in days
    IP := normalize( ( JD - 2451550.1 ) / 29.530588853 )
    AG := IP*29.53

    if    AG <  1.84566 then Phase := "NEW"
    elsif AG <  5.53699 then Phase := "Waxing crescent"
    elsif AG <  9.22831 then Phase := "First quarter"
    elsif AG < 12.91963 then Phase := "Waxing gibbous"
    elsif AG < 16.61096 then Phase := "FULL"
    elsif AG < 20.30228 then Phase := "Waning gibbous"
    elsif AG < 23.99361 then Phase := "Last quarter"
    elsif AG < 27.68493 then Phase := "Waning crescent"
    else                     Phase := "NEW"
    end if

    IP := IP*2*PI                           % Convert phase to radians

    % calculate moon's distance
    DP := 2*PI*normalize( ( JD - 2451562.2 ) / 27.55454988 )
    DI := 60.4 - 3.3*cos( DP ) - 0.6*cos( 2*IP - DP ) - 0.5*cos( 2*IP )
```

```
    % calculate moon's ecliptic latitude
    NP := 2*PI*normalize( ( JD - 2451565.2 ) / 27.212220817 )
    LA := 5.1*sin( NP )

    % calculate moon's ecliptic longitude
    RP := normalize( ( JD - 2451555.8 ) / 27.321582241 )
    LO := 360*RP + 6.3*sin( DP ) + 1.3*sin( 2*IP - DP ) + 0.7*sin( 2*IP )

    if    LO <  33.18 then Zodiac := "Pisces"
    elsif LO <  51.16 then Zodiac := "Aries"
    elsif LO <  93.44 then Zodiac := "Taurus"
    elsif LO < 119.48 then Zodiac := "Gemini"
    elsif LO < 135.30 then Zodiac := "Cancer"
    elsif LO < 173.34 then Zodiac := "Leo"
    elsif LO < 224.17 then Zodiac := "Virgo"
    elsif LO < 242.57 then Zodiac := "Libra"
    elsif LO < 271.26 then Zodiac := "Scorpio"
    elsif LO < 302.49 then Zodiac := "Sagittarius"
    elsif LO < 311.72 then Zodiac := "Capricorn"
    elsif LO < 348.58 then Zodiac := "Aquarius"
    else                   Zodiac := "Pisces"
    end if

    % display results
    put "phase         = ", Phase
    put "age           = ", round2( AG ), " days"
    put "distance      = ", round2( DI ), " earth radii"
    put "ecliptic"
    put " latitude     = ", round2( LA ), '°'
    put " longitude    = ", round2( LO ), '°'
    put "constellation = ", Zodiac

end procedure

% check for valid date
function isdayofmonth( year, month, day : int ) : boolean

    var daysofmonth : int

    if (month < 1) or (12 < month) then
        return false                        % invalid month
    end if

    case month of                           % get days in this month
    value 4,6,9,11:
        daysofmonth := 30                   % Apr, Jun, Sep, Nov
    value 2:
        daysofmonth := 28                   % Feb normal
        if year mod 4 = 0 then
            if not((year mod 100 = 0) and
                   (year mod 400 ~= 0)) then
                daysofmonth := 29           % Feb leap year
            end if
        end if
    value:
        daysofmonth := 31                   % other months
    end case

    return (0 < day) and (day <= daysofmonth)

end function

% round to 2 decimal places
function round2( x : real ) : real
    return ( round( 100*x )/100.0 )
end function

% normalize values to range 0...1
function normalize( v : real ) : real
    v := v - floor( v )
```

```
    if v < 0 then
        v := v + 1
    end if
    return v
end function
```

Sample output

```
Moon on 3/24/2004
phase         = Waxing crescent
age           = 3.31 days
distance      = 62.87 earth radii
ecliptic
 latitude     = -0.1°
 longitude    = 44.92°
constellation = Aries
```

References

1. Sky & Telescope, *Astronomical Computing*, April 1994
