Numerical Recipes Routines and Examples in BASIC

companion manual to
Numerical Recipes: The Art of Scientific Computing

Julien C. Sprott
University of Wisconsin - Madison

in association with **Numerical Recipes Software**

CAMBRIDGE
UNIVERSITY PRESS

Published by the Press Syndicate of the University of Cambridge
The Pitt Building, Trumpington Street, Cambridge CB2 1RP
40 West 20th Street, New York, NY 10011-4211, USA
10 Stamford Road, Oakleigh, Melbourne 3166, Australia

First published 1991
Reprinted 1992, 1994, 1996, 1998

Typeset in TEX

Library of Congress Cataloging-in-Publication Data

Sprott, Julien C.
Numerical recipes : routines and examples in BASIC / Julien C. Sprott.
p. cm.
"Comparison manual to Numerical recipes : the art of scientific computing."
"In association with Numerical Recipes Software."
Includes index.
ISBN 0-521-40689-7 paperback. - ISBN 0-521-40688-9 diskette.
1. Numerical analysis-Computer programs. 2. Science-Mathematics-Computer programs. 3. BASIC (Computer program language) I. Numerical Recipes Software (Firm) II. Numerical recipes. III. Title.
QA297.S68 1991 90-26323
519.4'02085'5133-dc20 CIP

ISBN 0-521-40689-7 paperback
ISBN 0-521-40688-9 diskette

Transferred to digital printing 2002

CONTENTS

Chapter 1: Preliminaries

The routines in Chapter 1 of *Numerical Recipes : The Art of Scientific Computing* are introductory and less general in purpose than those in the remainder of the book. This chapter's routines serve primarily to expose the book's notational conventions, to illustrate control structures, and perhaps to amuse. You may even find them useful, but we hope that you will use BADLUK for no serious purpose.

* * * *

Subroutine FLMOON calculates the phases of the moon, or more exactly, the Julian day and fraction thereof on which a given phase will occur or has occurred. The program D1R1 asks the present date and compiles a list of upcoming phases. We have compared the predictions to lunar tables, with happy results. Shown are the results of a test run, which you may replicate as a check. In this program, notice that we have set TZONE (the time zone) to 5.0 to signify the five-hour separation of the Eastern Standard time zone from Greenwich, England. Our parochial viewpoint requires you to use negative values of TZONE if you are east of Greenwich. The Julian day results are converted to calendar dates through the use of CALDAT, which appears later in the chapter. The fractional Julian day and time zone combine to form a correction that can possibly change the calendar date by one day.

```
    Date          Time(EST)     Phase
 1  9 1982          3 PM      full moon
 1 16 1982          7 PM      last quarter
 1 24 1982         11 PM      new moon
 2  1 1982         10 AM      first quarter
 2  8 1982          2 AM      full moon
 2 15 1982          3 PM      last quarter
 2 23 1982          4 PM      new moon
 3  2 1982          6 PM      first quarter
 3  9 1982          3 PM      full moon
 3 17 1982         12 AM      last quarter
 3 25 1982          5 AM      new moon
 4  1 1982          0 AM      first quarter
 4  8 1982          5 AM      full moon
 4 16 1982          7 AM      last quarter
 4 23 1982          4 PM      new moon
 4 30 1982          7 AM      first quarter
 5  7 1982          8 PM      full moon
 5 16 1982          0 AM      last quarter
 5 23 1982          0 AM      new moon
 5 29 1982          3 PM      first quarter
```

Here is the recipe FLMOON:

```
SUB FLMOON (N, NPH, JD&, FRAC)
```

Our programs begin with an introductory comment summarizing their purpose and explaining their calling sequence. This routine calculates the phases of the moon. Given an integer N and a code NPH for the phase desired (NPH= 0 for new moon, 1 for first quarter, 2 for full, 3 for last quarter), the routine returns the Julian Day Number JD&, and the fractional part of a day FRAC to be added to it, of the N^{th} such phase since January, 1900. Coordinated Universal Time is assumed.

```
RAD = .017453293#
C = N + NPH / 4!                          This is how we comment an individual line.
T = C / 1236.85
T2 = T ^ 2
AQ = 359.2242 + 29.105356# * C      You aren't really intended to understand this algorithm,
AM = 306.0253 + 385.816918# * C + .01073 * T2        but it does work!
JD& = 2415020 + 28 * N + 7 * NPH
XTRA = .75933 + 1.53058868# * C + (.0001178 - 1.55E-07 * T) * T2
IF NPH = 0 OR NPH = 2 THEN
  XTRA = XTRA + (.1734 - .000393 * T) * SIN(RAD * AQ) - .4068 * SIN(RAD * AM)
ELSEIF NPH = 1 OR NPH = 3 THEN
  XTRA = XTRA + (.1721 - .0004 * T) * SIN(RAD * AQ) - .628 * SIN(RAD * AM)
ELSE
  PRINT "NPH is unknown."
  EXIT SUB                                This is how we will indicate error conditions.
END IF
IF XTRA >= 0! THEN
  I = INT(XTRA)
ELSE
  I = INT(XTRA - 1!)
END IF
JD& = JD& + I
FRAC = XTRA - I
END SUB
```

A sample program using FLMOON is the following:

```
DECLARE SUB FLMOON (N!, NPH!, JD&, FRAC!)
DECLARE SUB CALDAT (JULIAN&, MM!, ID!, IYYY!)
DECLARE FUNCTION JULDAY& (IM!, ID!, IY!)

'PROGRAM D1R1
'Driver for routine FLMOON
CLS
TZONE = 5!
DIM PHASE$(4), TIMSTR$(2)
FOR I = 1 TO 4
  READ PHASE$(I)
NEXT I
DATA "new moon","first quarter","full moon","last quarter"
FOR I = 1 TO 2
  READ TIMSTR$(I)
NEXT I
DATA " AM"," PM"
PRINT "Date of the next few phases of the moon"
PRINT "Enter today's date (e.g. 1,31,1982)"
INPUT IM, ID, IY
```

```
PRINT
TIMZON = -TZONE / 24!
'Approximate number of full moons since January 1900
N = INT(12.37 * (IY - 1900 + (IM - .5) / 12!))
NPH = 2
J1& = JULDAY&(IM, ID, IY)
CALL FLMOON(N, NPH, J2&, FRAC)
N = INT(N + (J1& - J2&) / 28!)
PRINT "   Date", "  Time(EST)", " Phase"
FOR I = 1 TO 20
  CALL FLMOON(N, NPH, J2&, FRAC)
  IFRAC = CINT(24! * (FRAC + TIMZON))
  IF IFRAC < 0 THEN
    J2& = J2& - 1
    IFRAC = IFRAC + 24
  END IF
  IF IFRAC >= 12 THEN
    J2& = J2& + 1
    IFRAC = IFRAC - 12
  ELSE
    IFRAC = IFRAC + 12
  END IF
  IF IFRAC > 12 THEN
    IFRAC = IFRAC - 12
    ISTR = 2
  ELSE
    ISTR = 1
  END IF
  CALL CALDAT(J2&, IM, ID, IY)
  PRINT USING "##"; IM;
  PRINT USING "###"; ID;
  PRINT USING "#####"; IY;
  PRINT USING "#########"; IFRAC;
  PRINT TIMSTR$(ISTR); "     "; PHASE$(NPH + 1)
  IF NPH = 3 THEN
    NPH = 0
    N = N + 1
  ELSE
    NPH = NPH + 1
  END IF
NEXT I
END
```

Program **JULDAY**, our exemplar of the **IF** control structure, converts calendar dates to Julian dates. Not many people know the Julian date of their birthday or any other convenient reference point, for that matter. To remedy this, we offer a list of checkpoints, which appears at the end of this chapter as the file **DATES.DAT**. The program **D1R2** lists the Julian date for each historic event for comparison. Then it allows you to make your own choices for entertainment.

Here is the recipe **JULDAY**:

```
FUNCTION JULDAY& (MM, ID, IYYY)
```

In this routine **JULDAY&** returns the Julian Day Number which begins at noon of the calendar date specified by month **MM**, day **ID**, and year **IYYY**. Positive year signifies A.D.; negative, B.C. Remember that the year after 1 B.C. was 1 A.D.

```
IGREG& = 588829                          Gregorian Calendar was adopted on Oct. 15, 1582.
IF IYYY = 0 THEN PRINT "There is no Year Zero.": EXIT FUNCTION
IF IYYY < 0 THEN IYYY = IYYY + 1
IF MM > 2 THEN                           Here is an example of a block IF-structure.
  JY = IYYY
  JM = MM + 1
ELSE
  JY = IYYY - 1
  JM = MM + 13
END IF
JD& = INT(365.25 * JY) + INT(30.6001 * JM) + ID + 1720995
IF ID + 31 * (MM + 12 * IYYY) >= IGREG& THEN          Change to Gregorian Calendar?
  JA = INT(.01 * JY)
  JD& = JD& + 2 - JA + INT(.25 * JA)
END IF
JULDAY& = JD&
END FUNCTION
```

A sample program using **JULDAY** is the following:

```
DECLARE FUNCTION JULDAY& (IM!, ID!, IY!)

'PROGRAM D1R2
'Driver for JULDAY
CLS
DIM NAMQ$(12)
FOR I = 1 TO 12
  READ NAMQ$(I)
NEXT I
DATA "January","February","March","April","May","June","July","August"
DATA "September","October","November","December"
OPEN "DATES.DAT" FOR INPUT AS #1
LINE INPUT #1, DUM$
LINE INPUT #1, DUM$
N = VAL(DUM$)
PRINT "Month          Day  Year   Julian Day    Event"
PRINT
FOR I = 1 TO N
  LINE INPUT #1, DUM$
  IM = VAL(MID$(DUM$, 1, 2))
  ID = VAL(MID$(DUM$, 3, 3))
  IY = VAL(MID$(DUM$, 6, 5))
  TXT$ = MID$(DUM$, 11)
  PRINT NAMQ$(IM),
  PRINT USING "###"; ID;
  PRINT USING "######"; IY,
  PRINT USING "###########"; JULDAY&(IM, ID, IY);
  PRINT "     "; TXT$
NEXT I
CLOSE #1
```

```
PRINT
PRINT "Month,Day,Year (e.g. 1,13,1905)"
PRINT
FOR I = 1 TO 20
  PRINT "MM,DD,YYYY"
  INPUT IM, ID, IY
  IF IM < 0 THEN EXIT FOR
  PRINT "Julian Day: "; JULDAY&(IM, ID, IY)
  PRINT
NEXT I
END
```

The next program in *Numerical Recipes* is **BADLUK**, an infamous code that combines the best and worst instincts of man. We include no demonstration program for **BADLUK**, not just because we fear it, but also because it is self-contained, with sample results appearing in the text. The purpose of **BADLUK** is to find all dates Friday the Thirteenth on which the moon is full.

Here is the recipe BADLUK:

```
DECLARE SUB FLMOON (N!, NPH!, JD&, FRAC!)
DECLARE FUNCTION JULDAY& (IM!, ID!, IY!)

'PROGRAM BADLUK
CLS
TIMZON = -5! / 24                    Time zone −5 is Eastern Standard Time.
READ IYBEG, IYEND                    The range of dates to be searched.
DATA 1900,2000
PRINT "Full moons on Friday the 13th from"; IYBEG; "to"; IYEND
PRINT
FOR IYYY = IYBEG TO IYEND            Loop over each year,
  FOR IM = 1 TO 12                   and each month.
    JDAY& = JULDAY&(IM, 13, IYYY)    Is the thirteenth a Friday?
    IDWK = (JDAY& + 1) - 7 * INT((JDAY& + 1) / 7)
    IF IDWK = 5 THEN
      N = INT(12.37 * (IYYY - 1900 + (IM - .5) / 12!))
      This value N is a first approximation to how many full moons have occurred since 1900. We
      will feed it into the phase routine and adjust it up or down until we determine that our desired
      13th was or was not a full moon. The variable ICON signals the direction of adjustment.
      ICON = 0
      DO
        CALL FLMOON(N, 2, JD&, FRAC)          Get date of full moon N.
        IFRAC = CINT(24! * (FRAC + TIMZON))   Convert to hours in correct time zone.
        IF IFRAC < 0 THEN                     Convert from Julian Days beginning at noon
          JD& = JD& - 1                       to civil days beginning at midnight.
          IFRAC = IFRAC + 24
        END IF
        IF IFRAC > 12 THEN
          JD& = JD& + 1
          IFRAC = IFRAC - 12
        ELSE
          IFRAC = IFRAC + 12
        END IF
        IF JD& = JDAY& THEN           Did we hit our target day?
          PRINT IM; "/"; 13; "/"; IYYY
```

```
          PRINT "Full moon"; IFRAC; "hrs after midnight (EST)."
          PRINT
          EXIT DO                    Part of the break-structure, case of a match.
        ELSE                         Didn't hit it.
          IC = SGN(JDAY& - JD&)
          IF IC = -ICON THEN EXIT DO          Another break, case of no match.
          ICON = IC
          N = N + IC
        END IF
      LOOP
    END IF
  NEXT IM
NEXT IYYY
END
```

Chapter 1 closes with routine CALDAT, which illustrates no new points, but complements JULDAY by doing conversions from Julian day number to the month, day, and year on which the given Julian day began. This offers an opportunity, grasped by the demonstration program D1R4, to push dates through both JULDAY and CALDAT in succession, to see if they survive intact. This, of course, tests only your authors' ability to make mistakes backward as well as forward, but we hope you will share our optimism that correct results here speak well for both routines. (We have checked them a bit more carefully in other ways.)

Here is the recipe CALDAT:

```
SUB CALDAT (JULIAN&, MM, ID, IYYY)
      Inverse of the function JULDAY given above. Here JULIAN& is input as a Julian Day Number,
      and the routine outputs MM,ID, and IYYY as the month, day, and year on which the specified
      Julian Day started at noon.
IGREG& = 2299161                     Cross-over to Gregorian Calendar
IF JULIAN& >= IGREG& THEN            produces this correction,
  JALPHA& = INT(((JULIAN& - 1867216) - .25) / 36524.25)
  JA& = JULIAN& + 1 + JALPHA& - INT(.25 * JALPHA&)
ELSE                                 or else no correction.
  JA& = JULIAN&
END IF
JB& = JA& + 1524
JC& = INT(6680! + ((JB& - 2439870) - 122.1) / 365.25)
JD& = 365 * JC& + INT(.25 * JC&)
JE& = INT((JB& - JD&) / 30.6001)
ID = JB& - JD& - INT(30.6001 * JE&)
MM = JE& - 1
IF MM > 12 THEN MM = MM - 12
IYYY = JC& - 4715
IF MM > 2 THEN IYYY = IYYY - 1
IF IYYY <= 0 THEN IYYY = IYYY - 1
END SUB
```

A sample program using **CALDAT** is the following:

```
DECLARE SUB CALDAT (JULIAN&, MM!, ID!, IYYY!)
DECLARE FUNCTION JULDAY& (IM!, ID!, IY!)

'PROGRAM D1R4
'Driver for routine CALDAT
CLS
DIM NAMQ$(12)
'Check whether CALDAT properly undoes the operation of JULDAY
FOR I = 1 TO 12
  READ NAMQ$(I)
NEXT I
DATA "January","February","March","April","May","June","July","August"
DATA "September","October","November","December"
OPEN "DATES.DAT" FOR INPUT AS #1
LINE INPUT #1, DUM$
LINE INPUT #1, DUM$
N = VAL(DUM$)
PRINT "Original Date:                          Reconstructed Date:"
PRINT "Month     Day  Year    Julian Day     Month      Day  Year"
PRINT
FOR I = 1 TO N
  LINE INPUT #1, DUM$
  IM = VAL(MID$(DUM$, 1, 2))
  ID = VAL(MID$(DUM$, 3, 3))
  IY = VAL(MID$(DUM$, 6, 10))
  IYCOPY = IY
  J& = JULDAY&(IM, ID, IYCOPY)
  CALL CALDAT(J&, IMM, IDD, IYY)
  PRINT NAMQ$(IM); TAB(12);
  PRINT USING "##"; ID;
  PRINT USING "######"; IY;
  PRINT "     "; J&; "     "; NAMQ$(IMM); TAB(50);
  PRINT USING "##"; IDD;
  PRINT USING "######"; IYY
NEXT I
END
```

Appendix

File **DATES.DAT**:

```
List of dates for testing routines in Chapter 1
16 entries
12 31   -1 End of millennium
01 01    1 One day later
10 14 1582 Day before Gregorian calendar
10 15 1582 Gregorian calendar adopted
01 17 1706 Benjamin Franklin born
04 14 1865 Abraham Lincoln shot
04 18 1906 San Francisco earthquake
05 07 1915 Sinking of the Lusitania
07 20 1923 Pancho Villa assassinated
05 23 1934 Bonnie and Clyde eliminated
07 22 1934 John Dillinger shot
```

```
04 03 1936 Bruno Hauptmann electrocuted
05 06 1937 Hindenburg disaster
07 26 1956 Sinking of the Andrea Doria
06 05 1976 Teton dam collapse
05 23 1968 Julian Day 2440000
```