

NAG Library Routine Document

C06LAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details.

1 Purpose

C06LAF estimates values of the inverse Laplace transform of a given function using a Fourier series approximation. Real and imaginary parts of the function, and a bound on the exponential order of the inverse, are required.

2 Specification

```

SUBROUTINE C06LAF (FUN, N, T, VALINV, ERREST, RELERR, ALPHAB, TFAC,      &
                  MXTERM, NTERMS, NA, ALOW, AHIGH, NFEVAL, WORK, IFAIL)
INTEGER           N, MXTERM, NTERMS, NA, NFEVAL, IFAIL
REAL (KIND=nag_wp) T(N), VALINV(N), ERREST(N), RELERR, ALPHAB, TFAC,  &
                  ALOW, AHIGH, WORK(4*MXTERM+2)
EXTERNAL         FUN

```

3 Description

Given a function $F(p)$ defined for complex values of p , C06LAF estimates values of its inverse Laplace transform by Crump's method (see Crump (1976)). (For a definition of the Laplace transform and its inverse, see the C06 Chapter Introduction.)

Crump's method applies the epsilon algorithm (see Wynn (1956)) to the summation in Durbin's Fourier series approximation (see Durbin (1974))

$$f(t_j) \simeq \frac{e^{at_j}}{\tau} \left[\frac{1}{2}F(a) - \sum_{k=1}^{\infty} \left\{ \operatorname{Re} \left(F \left(a + \frac{k\pi i}{\tau} \right) \right) \cos \frac{k\pi t_j}{\tau} - \operatorname{Im} \left(F \left(a + \frac{k\pi i}{\tau} \right) \right) \sin \frac{k\pi t_j}{\tau} \right\} \right],$$

for $j = 1, 2, \dots, n$, by choosing a such that a prescribed relative error should be achieved. The method is modified slightly if $t = 0.0$ so that an estimate of $f(0.0)$ can be obtained when it has a finite value. τ is calculated as $t_{\text{fac}} \times \max(0.01, t_j)$, where $t_{\text{fac}} > 0.5$. You specify t_{fac} and α_b , an upper bound on the exponential order α of the inverse function $f(t)$. α has two alternative interpretations:

- (i) α is the smallest number such that

$$|f(t)| \leq m \times \exp(\alpha t)$$

for large t ,

- (ii) α is the real part of the singularity of $F(p)$ with largest real part.

The method depends critically on the value of α . See Section 9 for further details. The routine calculates at least two different values of the parameter a , such that $a > \alpha_b$, in an attempt to achieve the requested relative error and provide error estimates. The values of t_j , for $j = 1, 2, \dots, n$, must be supplied in monotonically increasing order. The routine calculates the values of the inverse function $f(t_j)$ in decreasing order of j .

4 References

Crump K S (1976) Numerical inversion of Laplace transforms using a Fourier series approximation *J. Assoc. Comput. Mach.* **23** 89–96

Durbin F (1974) Numerical inversion of Laplace transforms: An efficient improvement to Dubner and Abate's method *Comput. J.* **17** 371–376

Wynn P (1956) On a device for computing the $e_m(S_n)$ transformation *Math. Tables Aids Comput.* **10** 91–96

5 Parameters

- 1: FUN – SUBROUTINE, supplied by the user. *External Procedure*
 FUN must evaluate the real and imaginary parts of the function $F(p)$ for a given value of p .

The specification of FUN is:

```
SUBROUTINE FUN (PR, PI, FR, FI)
```

```
REAL (KIND=nag_wp) PR, PI, FR, FI
```

1: PR – REAL (KIND=nag_wp)

Input

2: PI – REAL (KIND=nag_wp)

Input

On entry: the real and imaginary parts of the argument p .

3: FR – REAL (KIND=nag_wp)

Output

4: FI – REAL (KIND=nag_wp)

Output

On exit: the real and imaginary parts of the value $F(p)$.

FUN must either be a module subprogram USED by, or declared as EXTERNAL in, the (sub)program from which C06LAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

- 2: N – INTEGER *Input*

On entry: n , the number of points at which the value of the inverse Laplace transform is required.

Constraint: $N \geq 1$.

- 3: T(N) – REAL (KIND=nag_wp) array *Input*

On entry: each $T(j)$ must specify a point at which the inverse Laplace transform is required, for $j = 1, 2, \dots, n$.

Constraint: $0.0 \leq T(1) < T(2) < \dots < T(n)$.

- 4: VALINV(N) – REAL (KIND=nag_wp) array *Output*

On exit: an estimate of the value of the inverse Laplace transform at $t = T(j)$, for $j = 1, 2, \dots, n$.

- 5: ERREST(N) – REAL (KIND=nag_wp) array *Output*

On exit: an estimate of the error in $VALINV(j)$. This is usually an estimate of relative error but, if $VALINV(j) < RELERR$, $ERREST(j)$ estimates the absolute error. $ERREST(j)$ is unreliable when $VALINV(j)$ is small but slightly greater than $RELERR$.

- 6: RELERR – REAL (KIND=nag_wp) *Input*

On entry: the required relative error in the values of the inverse Laplace transform. If the absolute value of the inverse is less than $RELERR$, then absolute accuracy is used instead. $RELERR$ must

be in the range $0.0 \leq \text{RELERR} < 1.0$. If RELERR is set too small or to 0.0, then the routine uses a value sufficiently larger than *machine precision*.

- 7: ALPHAB – REAL (KIND=nag_wp) Input
On entry: α_b , an upper bound for α (see Section 3). Usually, α_b should be specified equal to, or slightly larger than, the value of α . If $\alpha_b < \alpha$ then the prescribed accuracy may not be achieved or completely incorrect results may be obtained. If α_b is too large C06LAF will be inefficient and convergence may not be achieved.
Note: it is as important to specify α_b correctly as it is to specify the correct function for inversion.
- 8: TFAC – REAL (KIND=nag_wp) Input
On entry: t_{fac} , a factor to be used in calculating the parameter τ . Larger values (e.g., 5.0) may be specified for difficult problems, but these may require very large values of MXTERM.
Suggested value: TFAC = 0.8.
Constraint: TFAC > 0.5.
- 9: MXTERM – INTEGER Input
On entry: the maximum number of (complex) terms to be used in the evaluation of the Fourier series.
Suggested value: MXTERM \geq 100, except for very simple problems.
Constraint: MXTERM \geq 1.
- 10: NTERMS – INTEGER Output
On exit: the number of (complex) terms actually used.
- 11: NA – INTEGER Output
On exit: the number of values of a used by the routine. See Section 9.
- 12: ALOW – REAL (KIND=nag_wp) Output
On exit: the smallest value of a used in the algorithm. This may be used for checking the value of ALPHAB– see Section 9.
- 13: AHIGH – REAL (KIND=nag_wp) Output
On exit: the largest value of a used in the algorithm. This may be used for checking the value of ALPHAB– see Section 9.
- 14: NFEVAL – INTEGER Output
On exit: the number of calls to FUN made by the routine.
- 15: WORK($4 \times \text{MXTERM} + 2$) – REAL (KIND=nag_wp) array Workspace
- 16: IFAIL – INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.
 For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output parameters may be useful even if IFAIL \neq 0 on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Note: C06LAF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL = 1

On entry, $N < 1$,
 or $MXTERM < 1$,
 or $RELERR < 0.0$,
 or $RELERR \geq 1.0$,
 or $TFAC \leq 0.5$.

IFAIL = 2

On entry, $T(1) < 0.0$,
 or $T(1), T(2), \dots, T(N)$ are not in strictly increasing order.

IFAIL = 3

$T(N)$ is too large for this value of ALPHAB. If necessary, scale the problem as described in Section 9.

IFAIL = 4

The required accuracy cannot be obtained. It is possible that ALPHAB is less than α . Alternatively, the problem may be especially difficult. Try increasing TFAC, ALPHAB or both.

IFAIL = 5

Convergence failure in the epsilon algorithm. Some values of $VALINV(j)$ may be calculated to the desired accuracy; this may be determined by examining the values of $ERREST(j)$. Try reducing the range of T or increasing MXTERM. If IFAIL = 5 still results, try reducing TFAC.

IFAIL = 6

All values of $VALINV(j)$ have been calculated but not all are to the requested accuracy; the values of $ERREST(j)$ should be examined carefully. Try reducing the range of t , or increasing TFAC, ALPHAB or both.

IFAIL = -99

An unexpected error has been triggered by this routine. Please contact NAG.
 See Section 3.8 in the Essential Introduction for further information.

IFAIL = -399

Your licence key may have expired or may not have been installed correctly.
 See Section 3.7 in the Essential Introduction for further information.

IFAIL = -999

Dynamic memory allocation failed.
 See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

The error estimates are often very close to the true error but, because the error control depends on an asymptotic formula, the required error may not always be met. There are two principal causes of this: Gibbs' phenomena, and zero or small values of the inverse Laplace transform.

Gibbs' phenomena (see the C06 Chapter Introduction) are exhibited near $t = 0.0$ (due to the method) and around discontinuities in the inverse Laplace transform $f(t)$. If there is a discontinuity at $t = c$ then the method converges such that $f(c) \rightarrow (f(c-) + f(c+))/2$.

Apparent loss of accuracy, when $f(t)$ is small, may not be serious. Crump's method keeps control of relative error so that good approximations to small function values may appear to be very inaccurate. If $|f(t)|$ is estimated to be less than RELERR then this routine switches to absolute error estimation. However, when $|f(t)|$ is slightly larger than RELERR the relative error estimates are likely to cause IFAIL = 6. If this is found inconvenient it can sometimes be avoided by adding k/p to the function $F(p)$, which shifts the inverse to $k + f(t)$.

Loss of accuracy may also occur for highly oscillatory functions.

More serious loss of accuracy can occur if α is unknown and is incorrectly estimated. See Section 9.

8 Parallelism and Performance

Not applicable.

9 Further Comments

9.1 Timing

The value of n is less important in general than the value of NTERMS. Unless FUN is very inexpensive to compute, the timing is proportional to $NA \times NTERMS$. For simple problems $NA = 2$ but in difficult problems NA may be somewhat larger.

9.2 Precautions

You are referred to the C06 Chapter Introduction for advice on simplifying problems with particular difficulties, e.g., where the inverse is known to be a step function.

The method does not work well for large values of t when α is positive. It is advisable, especially if IFAIL = 3 is obtained, to scale the problem if $|\alpha|$ is much greater than 1.0. See the C06 Chapter Introduction.

The range of values of t specified for a particular call should not be greater than about 10 units. This is because the method uses parameters based on the value $T(n)$ and these tend to be less appropriate as t becomes smaller. However, as the timing of the routine is not especially dependent on n , it is usually far more efficient to evaluate the inverse for ranges of t than to make separate calls to the routine for each value of t .

The most important parameter to specify correctly is ALPHAB, an upper bound for α . If, on entry, ALPHAB is sufficiently smaller than α then completely incorrect results will be obtained with IFAIL = 0. Unless α is known theoretically it is strongly advised that you should test any estimated value used. This may be done by specifying a single value of t (i.e $T(n)$, $n = 1$) with two sets of suitable values of TFAC, RELERR and MXTERM, and examining the resulting values of ALOW and AHIGH. The value of $T(1)$ should be chosen very carefully and the following points should be borne in mind:

- (i) $T(1)$ should be small but not too close to 0.0 because of Gibbs' phenomenon (see Section 7),
- (ii) the larger the value of $T(1)$, the smaller the range of values of a that will be used in the algorithm,
- (iii) $T(1)$ should ideally not be chosen such that $f(T(1)) = 0.0$ or a very small value. For suitable problems $T(1)$ might be chosen as, say, 0.1 or 1.0 depending on these factors. The routine calculates ALOW from the formula

$$ALOW = ALPHAB - \frac{\ln(0.1 \times RELERR)}{2 \times \tau}$$

Additional values of a are computed by adding $1/\tau$ to the previous value. As $\tau = TFAC \times T(n)$, it will be seen that large values of TFAC and RELERR will test for a close to ALPHAB. Small values of TFAC and RELERR will test for a large. If the result of both tests is IFAIL = 0, with comparable values for the inverse, then this gives some credibility to the chosen value of ALPHAB. You should note that this test could be more computationally expensive than the calculation of the inverse itself. The example program (see Section 10) illustrates how such a test may be performed.

10 Example

This example estimates the inverse Laplace transform of the function $F(p) = 1/(p + 1/2)$. The true inverse of $F(p)$ is $\exp(-t/2)$. Two preliminary calls to the routine are made to verify that the chosen value of ALPHAB is suitable. For these tests the single value $T(1) = 1.0$ is used. To test values of a close to ALPHAB, the values TFAC = 5.0 and RELERR = 0.01 are chosen. To test larger a , the values TFAC = 0.8 and RELERR = 1.0E-3 are used. Because the values of the computed inverse are similar and IFAIL = 0 in each case, these tests show that there is unlikely to be a singularity of $F(p)$ in the region $-0.04 \leq \text{Re}(p) \leq 6.51$.

10.1 Program Text

```
!   C06LAF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

Module c06lafe_mod

!   C06LAF Example Program Module:
!   Parameters and User-defined Routines

!   .. Use Statements ..
Use nag_library, Only: nag_wp
!   .. Implicit None Statement ..
Implicit None
!   .. Accessibility Statements ..
Private
Public                                :: fun
!   .. Parameters ..
Integer, Parameter, Public           :: nin = 5, nout = 6
Contains
Subroutine fun(pr,pi,fr,fi)
!   Function to be inverted

!   .. Use Statements ..
Use nag_library, Only: a02acf
!   .. Scalar Arguments ..
Real (Kind=nag_wp), Intent (Out)     :: fi, fr
Real (Kind=nag_wp), Intent (In)     :: pi, pr
!   .. Local Scalars ..
Real (Kind=nag_wp)                   :: xi, xr, yi, yr
!   .. Executable Statements ..
xr = 1.0_nag_wp
xi = 0.0_nag_wp
yr = pr + 0.5_nag_wp
yi = pi

Call a02acf(xr,xi,yr,yi,fr,fi)

Return
End Subroutine fun
End Module c06lafe_mod

Program c06lafe

!   C06LAF Example Main Program
```

```

! .. Use Statements ..
Use nag_library, Only: c06laf, nag_wp
Use c06lafe_mod, Only: fun, nin, nout
! .. Implicit None Statement ..
Implicit None
! .. Local Scalars ..
Real (Kind=nag_wp)           :: ahigh, alow, alphab, relerr, tfac
Integer                      :: i, ifail, itest, mxterm, n, n1, &
                             na, nfeval, nterms
! .. Local Arrays ..
Real (Kind=nag_wp), Allocatable :: errest(:), t(:), trurel(:), &
trures(:), valinv(:), work(:)
! .. Intrinsic Procedures ..
Intrinsic                    :: abs, exp, real
! .. Executable Statements ..
Write (nout,*) 'C06LAF Example Program Results'
Write (nout,*)
Write (nout,*) '(results may be machine-dependent)'
! Skip heading in data file
Read (nin,*)
Read (nin,*) n, mxterm
Allocate (errest(n),t(n),trurel(n),trures(n),valinv(n),work(4*mxterm+2))

t(1) = 1.0_nag_wp
alphab = -0.5_nag_wp

tests: Do itest = 1, 3

    Select Case (itest)
    Case (1)
!       Test for values of a close to alphab
        relerr = 0.01E0_nag_wp
        tfac = 7.5E0_nag_wp
        n1 = 1
        Write (nout,99997) t(1)
    Case (2)
!       Test for larger values of a
        relerr = 1.0E-3_nag_wp
        tfac = 0.8E0_nag_wp
        n1 = 1
        Write (nout,99997) t(1)
    Case (3)
        Write (nout,'(/lx,A/)') 'Compute inverse'
        n1 = 5
        Do i = 1, n1
            t(i) = real(i,kind=nag_wp)
        End Do
    End Select
    Write (nout,99999) mxterm, tfac, alphab, relerr

    ifail = -1
    Call c06laf(fun,n1,t,valinv,errest,relerr,alphab,tfac,mxterm,nterms, &
na,alow,ahigh,nfeval,work,ifail)

    If (ifail==0 .Or. ifail>=5) Then
        Write (nout,*)
        Write (nout,*) '      T      Result      exp(-T/2)      ', &
'Relative error  Error estimate'
        trures(1:n1) = exp(-0.5_nag_wp*t(1:n1))
        trurel(1:n1) = abs((valinv(1:n1)-trures(1:n1))/trures(1:n1))
        Write (nout,99998)(t(i),valinv(i),trures(i),trurel(i),errest(i),i=1, &
n1)
    Else
        Exit tests
    End If
End Do tests
If (ifail>=0) Then
    Write (nout,99996) nterms, nfeval, alow, ahigh, ifail
End If

```

```

99999 Format (1X,' MXTERM =',I4,' TFAC =',F6.2,' ALPHAB =',F6.2, &
' RELERR =',1P,E8.1)
99998 Format (1X,F4.1,7X,F6.3,9X,F6.3,8X,E8.1,8X,E8.1)
99997 Format (/1X,'Test with T(1) =',F4.1/)
99996 Format (/1X,' NTERMS =',I4,' NFEVAL =',I4,' ALOW =',F7.2,' AHIGH =', &
F7.2,' IFAIL =',I2)
End Program c06lafa

```

10.2 Program Data

C06LAF Example Program Data
 20 200 : n, mxterm

10.3 Program Results

C06LAF Example Program Results

(results may be machine-dependent)

Test with T(1) = 1.0

MXTERM = 200 TFAC = 7.50 ALPHAB = -0.50 RELERR = 1.0E-02

T	Result	exp(-T/2)	Relative error	Error estimate
1.0	0.607	0.607	0.1E-02	0.4E-02

Test with T(1) = 1.0

MXTERM = 200 TFAC = 0.80 ALPHAB = -0.50 RELERR = 1.0E-03

T	Result	exp(-T/2)	Relative error	Error estimate
1.0	0.607	0.607	0.2E-04	0.8E-04

Compute inverse

MXTERM = 200 TFAC = 0.80 ALPHAB = -0.50 RELERR = 1.0E-03

T	Result	exp(-T/2)	Relative error	Error estimate
1.0	0.607	0.607	0.5E-04	0.3E-03
2.0	0.368	0.368	0.7E-05	0.9E-04
3.0	0.223	0.223	0.2E-04	0.8E-04
4.0	0.135	0.135	0.1E-04	0.8E-04
5.0	0.082	0.082	0.2E-04	0.8E-04

NTERMS = 23 NFEVAL = 43 ALOW = 0.65 AHIGH = 0.90 IFAIL = 0

