

NAG Fortran Library Routine Document

S01EAF

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

S01EAF evaluates the exponential function e^z , for *complex* z .

2 Specification

```
complex FUNCTION S01EAF(Z, IFAIL)
INTEGER          IFAIL
complex        Z
```

3 Description

This routine evaluates the exponential function e^z , taking care to avoid machine overflow, and giving a warning if the result cannot be computed to more than half precision. The function is evaluated as $e^z = e^x(\cos y + i \sin y)$, where x and y are the real and imaginary parts respectively of z .

Since $\cos y$ and $\sin y$ are less than or equal to 1 in magnitude, it is possible that e^x may overflow although $e^x \cos y$ or $e^x \sin y$ does not. In this case the alternative formula $\text{sign}(\cos y)e^{x+\ln|\cos y|}$ is used for the real part of the result, and $\text{sign}(\sin y)e^{x+\ln|\sin y|}$ for the imaginary part. If either part of the result still overflows, a warning is returned through parameter IFAIL.

If $\text{Im } z$ is too large, precision may be lost in the evaluation of $\sin y$ and $\cos y$. Again, a warning is returned through IFAIL.

4 References

None.

5 Parameters

1: Z – *complex* *Input*

On entry: the argument z of the function.

2: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

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, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

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).

Errors or warnings detected by the routine:

IFAIL = 1

The real part of the result overflows, and is set to the largest safe number with the correct sign. The imaginary part of the result is meaningful.

IFAIL = 2

The imaginary part of the result overflows, and is set to the largest safe number with the correct sign. The real part of the result is meaningful.

IFAIL = 3

Both real and imaginary parts of the result overflow, and are set to the largest safe number with the correct signs.

IFAIL = 4

The computed result is accurate to less than half precision, due to the size of $\text{Im } z$.

IFAIL = 5

The computed result has no precision, due to the size of $\text{Im } z$, and is set to zero.

7 Accuracy

Accuracy is limited in general only by the accuracy of the Fortran intrinsic functions in the computation of $\sin y$, $\cos y$ and e^x , where $x = \text{Re } z$, $y = \text{Im } z$. As y gets larger, precision will probably be lost due to argument reduction in the evaluation of the sine and cosine functions, until the warning error IFAIL = 4 occurs when y gets larger than $\sqrt{1/\epsilon}$, where ϵ is the *machine precision*. Note that on some machines, the intrinsic functions SIN and COS will not operate on arguments larger than about $\sqrt{1/\epsilon}$, and so IFAIL can never return as 4.

In the comparatively rare event that the result is computed by the formulae $\text{sign}(\cos y)e^{x+\ln|\cos y|}$ and $\text{sign}(\sin y)e^{x+\ln|\sin y|}$, a further small loss of accuracy may be expected due to rounding errors in the logarithmic function.

8 Further Comments

None.

9 Example

The example program reads values of the argument z from a file, evaluates the function at each value of z and prints the results.

9.1 Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*      S01EAF Example Program Text
*      Mark 14 Release.  NAG Copyright 1989.
*      .. Parameters ..
INTEGER          NIN, NOUT
PARAMETER        (NIN=5,NOUT=6)
*      .. Local Scalars ..
complex        W, Z
INTEGER          IFAIL
*      .. External Functions ..
complex        S01EAF
EXTERNAL         S01EAF
*      .. Executable Statements ..
WRITE (NOUT,*) 'S01EAF Example Program Results'
*      Skip heading in data file
READ (NIN,*)
WRITE (NOUT,*)
WRITE (NOUT,*)
+ '          Z                      exp(Z)'
20 READ (NIN,*,END=40) Z
   IFAIL = 0
*
   W = S01EAF(Z,IFAIL)
*
   WRITE (NOUT,99999) Z, W
   GO TO 20
40 STOP
*
99999 FORMAT (1X,'(',F12.4,',',F12.4,')',F12.4,',',F12.4,')')
END
```

9.2 Program Data

```
S01EAF Example Program Data
( 1.0, 0.0)
(-0.5, 2.0)
( 0.0,-2.0)
(-2.5,-1.5)
```

9.3 Program Results

S01EAF Example Program Results

Z		exp(Z)	
(1.0000,	0.0000)	(2.7183,	0.0000)
(-0.5000,	2.0000)	(-0.2524,	0.5515)
(0.0000,	-2.0000)	(-0.4161,	-0.9093)
(-2.5000,	-1.5000)	(0.0058,	-0.0819)
