# NAG Library Function Document

# nag_lambertW_complex (c05bbc)

## 1 Purpose

nag_lambertW_complex (c05bbc) computes the values of Lambert's $W$ function $W(z)$.

## 2 Specification

```
#include <nag.h>
#include <nagc05.h>
void nag_lambertW_complex (Integer branch, Nag_Boolean offset, Complex z,
     Complex *w, double *resid, NagError *fail)
```

## 3 Description

nag_lambertW_complex (c05bbc) calculates an approximate value for Lambert's $W$ function (sometimes known as the 'product log' or 'Omega' function), which is the inverse function of

$$f(w) = we^w \quad \text{for} \quad w \in C.$$

The function $f$ is many-to-one, and so, except at 0, $W$ is multivalued. nag_lambertW_complex (c05bbc) allows you to specify the branch of $W$ on which you would like the results to lie by using the argument **branch**. Our choice of branch cuts is as in Corless *et al.* (1996), and the ranges of the branches of $W$ are summarised in Figure 1.

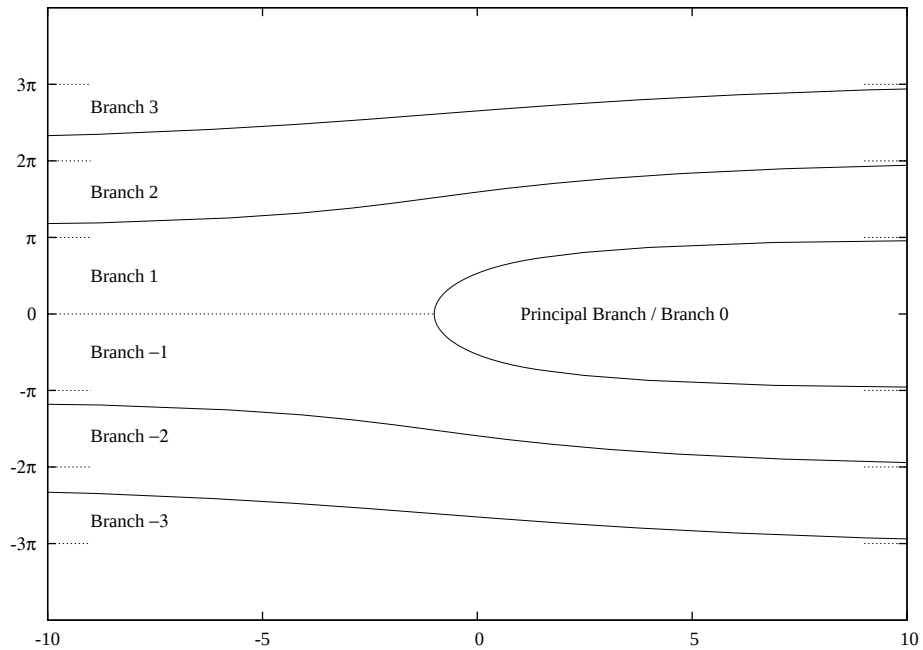


**Figure 1**
Ranges of the branches of $W(z)$

For more information about the closure of each branch, which is not displayed in Figure 1, see Corless *et al.* (1996). The dotted lines in the Figure denote the asymptotic boundaries of the branches, at multiples of $\pi$.

The precise method used to approximate $W$ is as described in Corless *et al.* (1996). For $z$ close to $-\exp(-1)$ greater accuracy comes from evaluating $W(-\exp(-1)+\Delta z)$ rather than $W(z)$: by setting

**offset** = Nag_TRUE on entry you inform nag_lambertW_complex (c05bbc) that you are providing $\Delta z$, not $z$, in **z**.

## 4 References

Corless R M, Gonnet G H, Hare D E G, Jeffrey D J and Knuth D E (1996) On the Lambert $W$ function *Advances in Comp. Math.* **3** 329–359

## 5 Arguments

1: **branch** – Integer *Input*

*On entry*: the branch required.

2: **offset** – Nag_Boolean *Input*

*On entry*: controls whether or not **z** is being specified as an offset from $-\exp(-1)$.

3: **z** – Complex *Input*

*On entry*: if **offset** = Nag_TRUE, **z** is the offset $\Delta z$ from $-\exp(-1)$ of the intended argument to $W$; that is, $W(\beta)$ is computed, where $\beta = -\exp(-1) + \Delta z$.

If **offset** = Nag_FALSE, **z** is the argument $z$ of the function; that is, $W(\beta)$ is computed, where $\beta = z$.

4: **w** – Complex * *Output*

*On exit*: the value $W(\beta)$: see also the description of **z**.

5: **resid** – double * *Output*

*On exit*: the residual $|W(\beta)\exp(W(\beta)) - \beta|$: see also the description of **z**.

6: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

**NE_BAD_PARAM**

On entry, argument ⟨*value*⟩ had an illegal value.

**NE_INTERNAL_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

**NW_REAL**

For the given offset **z**, $W$ is negligibly different from $-1$: $\mathrm{Re}(\mathbf{z}) = \langle value \rangle$ and $\mathrm{Im}(\mathbf{z}) = \langle value \rangle$.

**z** is close to $-\exp(-1)$. Enter **z** as an offset to $-\exp(-1)$ for greater accuracy: $\mathrm{Re}(\mathbf{z}) = \langle value \rangle$ and $\mathrm{Im}(\mathbf{z}) = \langle value \rangle$.

**NW_TOO_MANY_ITER**

The iterative procedure used internally did not converge in $\langle value \rangle$ iterations. Check the value of **resid** for the accuracy of **w**.

## 7 Accuracy

For a high percentage of **z**, nag_lambertW_complex (c05bbc) is accurate to the number of decimal digits of precision on the host machine (see nag_decimal_digits (X02BEC)). An extra digit may be lost on some platforms and for a small proportion of **z**. This depends on the accuracy of the base-10 logarithm on your system.

## 8 Parallelism and Performance

nag_lambertW_complex (c05bbc) is not threaded in any implementation.

## 9 Further Comments

The following figures show the principal branch of $W$.

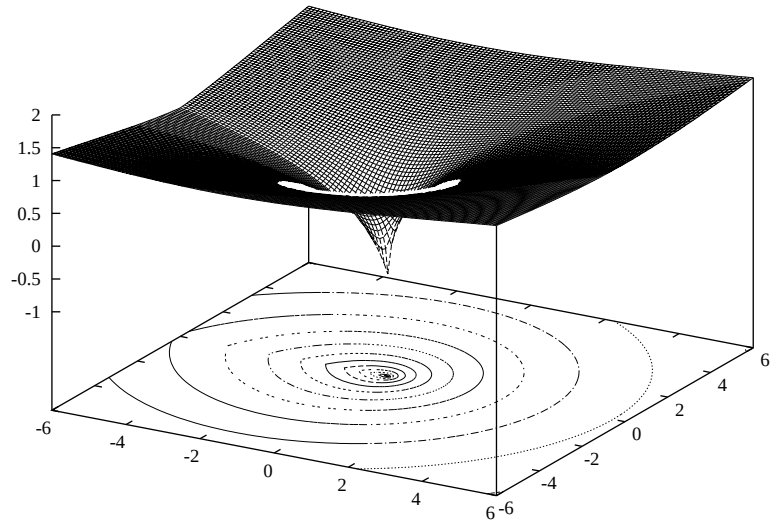


**Figure 2**
$\mathrm{real}(W_0(z))$

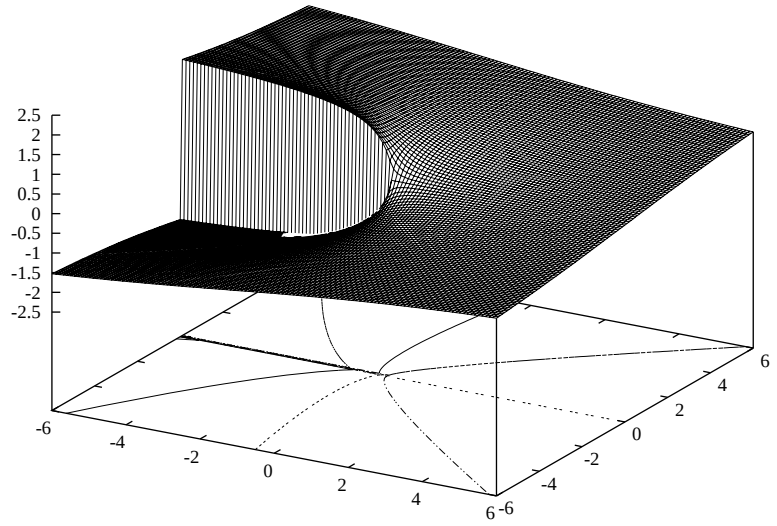


**Figure 3**
$\mathrm{Im}(W_0(z))$

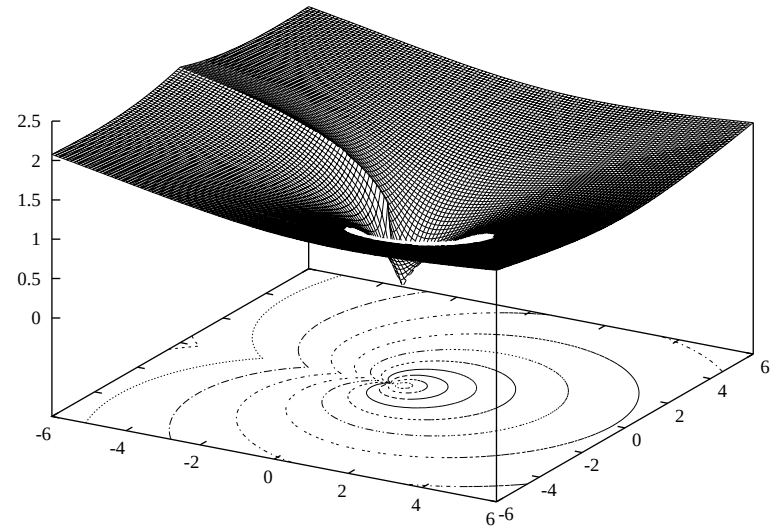


**Figure 4**
$\mathrm{abs}(W_0(z))$

## 10 Example

This example reads from a file the value of the required branch, whether or not the arguments to $W$ are to be considered as offsets to $-\exp(-1)$, and the arguments **z** themselves. It then evaluates the function for these sets of input data **z** and prints the results.

### 10.1 Program Text

```
/* nag_lambertW_complex (c05bbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <math.h>
#include <nag.h>
```

```
#include <nagx04.h>
#include <nag_stdlib.h>
#include <nagc05.h>

int main(void)
{
  /* Scalars */
  Complex w, z;
  double resid;
  Integer branch;
  Integer exit_status = 0;
  char offset[10];
  Nag_Boolean offsetenum;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_lambertW_complex (c05bbc) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &branch);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &branch);
#endif
#ifdef _WIN32
  scanf_s("%9s%*[^\n] ", offset, (unsigned)_countof(offset));
#else
  scanf("%9s%*[^\n] ", offset);
#endif

  /*
   * nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  offsetenum = (Nag_Boolean) nag_enum_name_to_value(offset);

  printf("\n");
  printf("branch = %" NAG_IFMT "\n", branch);
  printf("offset = %s\n", offset);
  printf("\n              z                              w(z)"
         "                  resid\n\n");
#ifdef _WIN32
  while (scanf_s(" (%lf,%lf)%*[^\n] ", &z.re, &z.im) != EOF)
#else
  while (scanf(" (%lf,%lf)%*[^\n] ", &z.re, &z.im) != EOF)
#endif
  {
    /*
     * nag_lambertW_complex (c05bbc)
     * Values of Lambert's W function, W(z)
     */
    nag_lambertW_complex(branch, offsetenum, z, &w, &resid, &fail);
    if (fail.code == NE_NOERROR ||
        fail.code == NW_REAL || fail.code == NW_TOO_MANY_ITER) {
      printf("(%14.5e,%14.5e) (%14.5e,%14.5e) %14.5e\n",
             z.re, z.im, w.re, w.im, resid);
    }
    else {
      printf("Error from nag_lambertW_complex (c05bbc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
```

```
  }

END:
  return exit_status;
}
```

## 10.2 Program Data

```
nag_lambertW_complex (c05bbc) Example Program Data
  0                                                     : branch
  Nag_FALSE                                             : offset
  (0.5, -1.0)
  (1.0, 2.3)
  (4.5, -0.1)
  (6.0, 6.0)
```

## 10.3 Program Results

```
nag_lambertW_complex (c05bbc) Example Program Results

branch = 0
offset = Nag_FALSE

               z                              w(z)                    resid

(   5.00000e-01,  -1.00000e+00) (   5.16511e-01,  -4.22053e-01)    5.55112e-17
(   1.00000e+00,   2.30000e+00) (   8.73606e-01,   5.76978e-01)    1.11022e-16
(   4.50000e+00,  -1.00000e-01) (   1.26735e+00,  -1.24194e-02)    0.00000e+00
(   6.00000e+00,   6.00000e+00) (   1.61492e+00,   4.90515e-01)    1.25607e-15
```