NAG Library Function Document

nag_complex (a02bac)

1 Purpose

nag_complex (a02bac) returns a complex number from real and imaginary parts.

2 Specification

```
#include <nag.h>
#include <naga02.h>
Complex nag_complex (double x, double y)
```

3 Description

None.

4 References

None.

5 Arguments

1: **x** – double *Input*

On entry: real part of complex number.

2: **y** – double *Input*

On entry: imaginary part of complex number.

6 Error Indicators and Warnings

None.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example illustrates the calls to all the complex functions in Chapter a02.

10.1 Program Text

```
/* nag_complex (a02bac) Example Program.
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <naga02.h>

int main(void)
{
  Integer     exit_status = 0;
  Complex     v, w, z;
  double      r, theta, x, y;
  Nag_Boolean equal, not_equal;

  printf("nag_complex (a02bac) Example Program Results\n");

  x = 2.0;
  y = -3.0;
  /* nag_complex (a02bac).
   * Complex number from real and imaginary parts
   */
  z = nag_complex(x, y);

  printf("  %-21s  %s    %8s =  %7.4f, %7.4f\n", "", "", "x, y",
          x, y);
  printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex",
          "z", "(x,y)", z.re, z.im);
  /* nag_complex_real (a02bbc).
   * Real part of a complex number
   */
  printf("  %-21s: %s    %8s =  %7.4f\n", "nag_complex_real",
          "", "real(z)", nag_complex_real(z));
  /* nag_complex_imag (a02bcc).
   * Imaginary part of a complex number
   */
  printf("  %-21s: %s    %8s =  %7.4f\n", "nag_complex_imag",
          "", "imag(z)", nag_complex_imag(z));
  /* nag_complex (a02bac), see above. */
  v = nag_complex(3.0, 1.25);
  /* nag_complex (a02bac), see above. */
  w = nag_complex(2.5, -1.75);
  printf("  %-21s: %s    %8s = (%7.4f, %7.4f)\n", "nag_complex", "",
          "v", v.re, v.im);
  printf("  %-21s: %s    %8s = (%7.4f, %7.4f)\n", "nag_complex", "",
          "w", w.re, w.im);
  /* nag_complex_add (a02cac).
   * Addition of two complex numbers
   */
  z = nag_complex_add(v, w);
  printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_add",
          "z", "v+w", z.re, z.im);
  /* nag_complex_subtract (a02cbc).
   * Subtraction of two complex numbers
   */
  z = nag_complex_subtract(v, w);
  printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n",
          "nag_complex_subtract", "z", "v-w", z.re, z.im);
  /* nag_complex_multiply (a02ccc).
   * Multiplication of two complex numbers
```

```
 */
z = nag_complex_multiply(v, w);
printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n",
        "nag_complex_multiply", "z", "v*w", z.re, z.im);
/* nag_complex_divide (a02cdc).
 * Quotient of two complex numbers
 */
z = nag_complex_divide(v, w);
printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_divide",
        "z", "v/w", z.re, z.im);
/* nag_complex_negate (a02cec).
 * Negation of a complex number
 */
z = nag_complex_negate(w);
printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_negate",
        "z", "-w", z.re, z.im);
/* nag_complex_conjg (a02cfc).
 * Conjugate of a complex number
 */
z = nag_complex_conjg(w);
printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_conjg",
        "z", "conjg(w)", z.re, z.im);
/* nag_complex_equal (a02cgc).
 * Equality of two complex numbers
 */
equal = nag_complex_equal(v, w);
if (equal)
  printf("  %-21s: %s == %s\n", "nag_complex_equal", "v", "w");
else
  printf("  %-21s: %s != %s\n", "nag_complex_equal", "v", "w");
/* nag_complex_not_equal (a02chc).
 * Inequality of two complex numbers
 */
not_equal = nag_complex_not_equal(w, z);
if (not_equal)
  printf("  %-21s: %s != %s\n\n", "nag_complex_not_equal", "w", "z");
else
  printf("  %-21s: %s == %s\n\n", "nag_complex_not_equal", "w", "z");

/* nag_complex_arg (a02dac).
 * Argument of a complex number
 */
theta = nag_complex_arg(z);
printf("  %-21s: %s    %8s =  %7.4f\n", "nag_complex_arg", "",
        "arg(z)", theta);
/* nag_complex_abs (a02dbc).
 * Modulus of a complex number
 */
r = nag_complex_abs(z);
printf("  %-21s: %s = %8s =  %7.4f\n", "nag_complex_abs", "r",
        "abs(z)", r);
/* nag_complex_sqrt (a02dcc).
 * Square root of a complex number
 */
v = nag_complex_sqrt(z);
printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_sqrt",
        "v", "sqrt(z)", v.re, v.im);
/* nag_complex_i_power (a02ddc).
 * Complex number raised to integer power
 */
v = nag_complex_i_power(z, (Integer) 3);
printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_i_power",
        "v", "z**3", v.re, v.im);
/* nag_complex_r_power (a02dec).
 * Complex number raised to real power
 */
v = nag_complex_r_power(z, 2.5);
printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_r_power",
        "v", "z**2.5", v.re, v.im);
/* nag_complex_c_power (a02dfc).
 * Complex number raised to complex power
```

```
   */
  v = nag_complex_c_power(z, w);
  printf("  %-21s: %s = %8s = (%7.4f,%8.4f)\n", "nag_complex_c_power",
          "v", "z**w", v.re, v.im);
  /* nag_complex_log (a02dgc).
   * Complex logarithm
   */
  v = nag_complex_log(z);
  printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_log",
          "v", "log(z)", v.re, v.im);
  /* nag_complex_exp (a02dhc).
   * Complex exponential
   */
  z = nag_complex_exp(v);
  printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_exp",
          "z", "exp(v)", z.re, z.im);
  /* nag_complex_sin (a02djc).
   * Complex sine
   */
  v = nag_complex_sin(z);
  printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_sin",
          "v", "sin(z)", v.re, v.im);
  /* nag_complex_cos (a02dkc).
   * Complex cosine
   */
  v = nag_complex_cos(z);
  printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_cos",
          "v", "cos(z)", v.re, v.im);
  /* nag_complex_tan (a02dlc).
   * Complex tangent
   */
  v = nag_complex_tan(z);
  printf("  %-21s: %s = %8s = (%7.4f, %7.4f)\n", "nag_complex_tan",
          "v", "tan(z)", v.re, v.im);
  /* nag_complex_divide (a02cdc), see above. */
  v = nag_complex_divide(nag_complex_sin(z), nag_complex_cos(z));
  printf("  %-21s:%13s = (%7.4f, %7.4f)\n", "nag_complex_divide",
          "sin(z)/cos(z)", v.re, v.im);

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_complex (a02bac) Example Program Results
                                x, y =   2.0000, -3.0000
  nag_complex          : z =    (x,y) = ( 2.0000, -3.0000)
  nag_complex_real     :      real(z) =   2.0000
  nag_complex_imag     :      imag(z) =  -3.0000
  nag_complex          :            v = ( 3.0000,  1.2500)
  nag_complex          :            w = ( 2.5000, -1.7500)
  nag_complex_add      : z =      v+w = ( 5.5000, -0.5000)
  nag_complex_subtract : z =      v-w = ( 0.5000,  3.0000)
  nag_complex_multiply : z =      v*w = ( 9.6875, -2.1250)
  nag_complex_divide   : z =      v/w = ( 0.5705,  0.8993)
  nag_complex_negate   : z =       -w = (-2.5000,  1.7500)
  nag_complex_conjg    : z = conjg(w) = ( 2.5000,  1.7500)
  nag_complex_equal    : v != w
  nag_complex_not_equal: w != z

  nag_complex_arg      :       arg(z) =   0.6107
  nag_complex_abs      : r =   abs(z) =   3.0516
  nag_complex_sqrt     : v =  sqrt(z) = ( 1.6661,  0.5252)
  nag_complex_i_power  : v =     z**3 = (-7.3438, 27.4531)
  nag_complex_r_power  : v =   z**2.5 = ( 0.7153, 16.2522)
  nag_complex_c_power  : v =     z**w = (43.1428,-19.5581)
```

```
 nag_complex_log      : v =   log(z) = ( 1.1157,  0.6107)
 nag_complex_exp      : z =   exp(v) = ( 2.5000,  1.7500)
 nag_complex_sin      : v =   sin(z) = ( 1.7740, -2.2355)
 nag_complex_cos      : v =   cos(z) = (-2.3747, -1.6700)
 nag_complex_tan      : v =   tan(z) = (-0.0569,  0.9814)
 nag_complex_divide   :sin(z)/cos(z) = (-0.0569,  0.9814)
```