NAG Library Function Document

nag_prob_vavilov (g01euc)

1 Purpose

nag_prob_vavilov (g01euc) returns the value of the Vavilov distribution function \( \Phi_V(\lambda; \kappa, \beta^2) \).

It is intended to be used after a call to nag_init_vavilov (g01zuc).

2 Specification

```c
#include <nag.h>
#include <nagg01.h>
double nag_prob_vavilov (double x, const double comm_arr[])
```

3 Description

nag_prob_vavilov (g01euc) evaluates an approximation to the Vavilov distribution function \( \Phi_V(\lambda; \kappa, \beta^2) \)
given by

\[
\Phi_V(\lambda; \kappa, \beta^2) = \int_\infty^{\lambda} \phi_V(\lambda; \kappa, \beta^2) \, d\lambda,
\]

where \( \phi(\lambda) \) is described in nag_prob_density_vavilov (g01muc). The method used is based on Fourier expansions. Further details can be found in Schorr (1974).

4 References


5 Arguments

1: \( x \) – double

\( \text{Input} \)

\( \text{On entry:} \) the argument \( \lambda \) of the function.

2: \( \text{comm_arr}[322] \) – const double

\( \text{Communication Array} \)

\( \text{On entry:} \) this must be the same argument \( \text{comm_arr} \) as returned by a previous call to nag_init_vavilov (g01zuc).

6 Error Indicators and Warnings

None.

7 Accuracy

At least five significant digits are usually correct.

8 Parallelism and Performance

Not applicable.
9 Further Comments

nag_prob_vavilov (g01euc) can be called repeatedly with different values of $\lambda$ provided that the values of $\kappa$ and $\beta^2$ remain unchanged between calls. Otherwise, nag_init_vavilov (g01zuc) must be called again. This is illustrated in Section 10.

10 Example

This example evaluates $F_Y(\lambda;\kappa,\beta^2)$ at $\lambda = 0.1$, $\kappa = 2.5$ and $\beta^2 = 0.7$, and prints the results.

10.1 Program Text

```c
/* nag_prob_vavilov (g01euc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 7, 2002. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>
#include <nagx02.h>

int main(void)
{
    /* Scalars */
    double c1, c2, x, rkappa, beta2, x1, xu, y;
    Integer exit_status, mode;
    NagError fail;

#define WKMAX 322
    double comm_arr[WKMAX];
    mode = 1;
    INIT_FAIL(fail);
    exit_status = 0;

    /* nag_real_largest_number (x02alc). * The largest positive model number */
    c1 = -nag_real_largest_number;
    /* nag_real_largest_number (x02alc), see above. */
    c2 = -nag_real_largest_number;

    printf(" nag_prob_vavilov (g01euc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef WIN32
    scanf_s("%*[\n]");
    #else
    scanf("%*[\n]");
    #endif

    #ifdef WIN32
    while (scanf_s("%lf%lf%lf%*[\n]", &x, &rkappa, &beta2) != EOF)
    #else
    while (scanf("%lf%lf%lf%*[\n]", &x, &rkappa, &beta2) != EOF)
    #endif
    {
        if ((rkappa != c1) || (beta2 != c2))
            /* nag_init_vavilov (g01zuc). */
```
* Initialization function for nag_prob_density_vavilov
  *(g01muc) and nag_prob_vavilov (g01euc)*
  
  nag_init_vavilov(rkappa, beta2, mode, &xl, &xu, comm_arr, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_init_vavilov (g01zuc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  
  /* nag_prob_vavilov (g01euc).
  * Vavilov distribution function
  * Phi_V((lambda; kappa)beta^2)
  */
  
y = nag_prob_vavilov(x, comm_arr);
  printf(" X Rkappa Beta2 Y\n\n");
  printf(" %3.1f %3.1f %3.1f %13.4e\n", x, rkappa, beta2, y);
  c1 = rkappa;
  c2 = beta2;
  
  END:
  return exit_status;
}

10.2 Program Data

nag_prob_vavilov (g01euc) Example Program Data
0.1 2.5 0.7 : Values of X, RKAPPA and BETA2

10.3 Program Results

nag_prob_vavilov (g01euc) Example Program Results

<table>
<thead>
<tr>
<th>X</th>
<th>Rkappa</th>
<th>Beta2</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2.5</td>
<td>0.7</td>
<td>9.9982e-01</td>
</tr>
</tbody>
</table>