**Purpose**

nag_init_vavilov (g01zuc) is used to initialize functions nag_prob_vavilov (g01euc) and nag_prob_density_vavilov (g01muc).

It is intended to be used before a call to nag_prob_vavilov (g01euc) or nag_prob_density_vavilov (g01muc).

**Specification**

```c
#include <nag.h>
#include <nagg01.h>
void nag_init_vavilov (double rkappa, double beta2, Integer mode, double *xl, double *xu, double comm_arr[], NagError *fail)
```

**Description**

nag_init_vavilov (g01zuc) initializes the array `comm_arr` for use by nag_prob_vavilov (g01euc) or nag_prob_density_vavilov (g01muc) in the evaluation of the Vavilov functions $\phi_V(\lambda; \kappa, \beta^2)$ and $\Phi_V(\lambda; \kappa, \beta^2)$ respectively.

Multiple calls to nag_prob_vavilov (g01euc) or nag_prob_density_vavilov (g01muc) can be made following a single call to nag_init_vavilov (g01zuc), provided that `rkappa` or `beta2` do not change, and that either all calls are to nag_prob_vavilov (g01euc) or all calls are to nag_prob_density_vavilov (g01muc). If you wish to call both nag_prob_vavilov (g01euc) and nag_prob_density_vavilov (g01muc), then you will need to initialize both separately.

**References**


**Arguments**

1. `rkappa` – double
   
   *Input*
   
   *On entry:* the argument $\kappa$ of the function.
   
   *Constraint:* $0.01 \leq \text{rkappa} \leq 10.0$.

2. `beta2` – double
   
   *Input*
   
   *On entry:* the argument $\beta^2$ of the function.
   
   *Constraint:* $0.0 \leq \text{beta2} \leq 1.0$.

3. `mode` – Integer
   
   *Input*
   
   *On entry:* if `mode = 0`, then nag_prob_density_vavilov (g01muc) is to be called after the call to nag_init_vavilov (g01zuc). Otherwise, nag_prob_vavilov (g01euc) is to be called.
Output

4: \texttt{xl} \text{ – double} * \\
On exit: \( x_l \), a threshold value below which \( \phi_V(\lambda;\kappa, \beta^2) \) will be set to zero by \texttt{nag_prob_density_vavilov} (g01muc) and \( \Phi_V(\lambda;\kappa, \beta^2) \) will be set to zero by \texttt{nag_prob_vavilov} (g01euc) if \( \lambda < x_l \).

Output

5: \texttt{xu} \text{ – double} * \\
On exit: \( x_u \), a threshold value above which \( \phi_V(\lambda;\kappa, \beta^2) \) will be set to zero by \texttt{nag_prob_density_vavilov} (g01muc) and \( \Phi_V(\lambda;\kappa, \beta^2) \) will be set to unity by \texttt{nag_prob_vavilov} (g01euc) if \( \lambda > x_u \).

Communication Array

6: \texttt{comm_arr[322]} \text{ – double} \\
On exit: this argument should be passed unchanged to \texttt{nag_prob_vavilov} (g01euc) or \texttt{nag_prob_density_vavilov} (g01muc).

Input/Output

7: \texttt{fail} \text{ – NagError} * \\
The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_ALLOC_FAIL}
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}
On entry, argument \textit{value} had an illegal value.

\textbf{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 3.6.6 in the Essential Introduction for further information.

\textbf{NE_NO_LICENCE}
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

\textbf{NE_REAL}
On entry, \texttt{beta2} = \textit{value}. \\
Constraint: \texttt{beta2} \leq 1.0.
On entry, \texttt{beta2} = \textit{value}. \\
Constraint: \texttt{beta2} \geq 0.0.
On entry, \texttt{rkappa} = \textit{value}. \\
Constraint: \texttt{rkappa} \leq 10.0.
On entry, \texttt{rkappa} = \textit{value}. \\
Constraint: \texttt{rkappa} \geq 0.01.

7 Accuracy
At least five significant digits are usually correct.
8 Parallelism and Performance
Not applicable.

9 Further Comments
None.

10 Example
See Section 10 in nag_prob_density_vavilov (g01muc) and nag_prob_vavilov (g01euc).