NAG Library Function Document

nag_opt_free (e04xzc)

1 Purpose

nag_opt_free (e04xzc) is the function for freeing memory allocated by a NAG C Library function to the e04 options structure, type Nag_E04_Opt. The function will only free memory which has been allocated to pointers within the options structure by an optimization function; it will not free memory you have allocated. The standard C function `free()` must **not** be used for freeing NAG allocated memory in Chapter e04.

2 Specification

```
#include <nag.h>
#include <nage04.h>
void nag_opt_free (Nag_E04_Opt *options, const char *p_name, NagError *fail)
```

3 Description

The optimization functions of Chapter e04 have a number of optional arguments, which are set by means of a structure of type Nag_E04_Opt. Optional argument values can be assigned to members of the options structure directly in the program text and/or by supplying the optional values in a file to be read by the function nag_opt_read (e04xyc).

Many of the optimization functions use pointers within the options structure as arrays. The appropriate amount of memory for the arrays will be allocated internally by the optimization function being used. The same options structure may be used in several calls to an optimization function: NAG allocated memory will be automatically freed and reallocated on each call to the optimization function. **This is the recommended method of use of the pointers within the options structure**.

If users wish to free NAG allocated memory from the options structure at any point in their program, then nag_opt_free (e04xzc) **must** be used to perform the freeing operation.

Memory may be allocated to the pointers in the options structure if the NAG default memory allocation is not wanted — nag_opt_free (e04xzc) will not free this user allocated memory. Dynamic memory allocated by you should be freed by the standard C library function `free()`. If it is intended to re-enter a NAG optimization function after this use of `free()`, with the intention of using the NAG default memory allocation, then the pointer involved **must** be set to **NULL** before re-entry.

The purpose of using nag_opt_free (e04xzc) to free NAG allocated memory instead of `free()` is to allow the optimization functions to maintain knowledge of which pointers have been allocated memory by a NAG function and you have allocated memory. If nag_opt_free (e04xzc) is not used to free the NAG allocated memory and the standard C function `free()` is used instead then there is the danger that any memory which is dynamically allocated will be freed by the optimization function.

To conserve memory nag_opt_free (e04xzc) should also be used to free NAG allocated memory within the options structure when that memory is no longer required, e.g., before returning from the function which calls the NAG C Library Chapter e04 functions. Any memory not freed will, of course, be freed when your program terminates.

4 References

None.

5 Arguments

1: **options** – Nag_E04_Opt * *Input/Output*

On entry: the options structure that was used in a call to an optimization function in Chapter e04. The pointers within the structure may have either NAG allocated memory or user allocated memory.

On exit: those pointers selected (see argument **p_name**) which pointed to NAG allocated memory will have been freed and set to **NULL**. Any user allocated memory will not be freed.

2: **p_name** – const char * *Input*

On entry: a character string specifying which pointer is to be freed. The string should give the optional argument or structure member name. If you wish to free all NAG allocated memory then an empty string `""` or the string `"all"` should be given. Please note that **p_name** is case sensitive and as such upper-case letters should not be used unless explicitly required.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_STR_UNKNOWN

String supplied, $\langle string \rangle$, does not match name of any pointer in the options structure.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

See Section 10 in nag_opt_lsq_no_deriv (e04fcc), nag_opt_lsq_deriv (e04gbc), nag_opt_lp (e04mfc), nag_opt_qp (e04nfc), nag_opt_nlp_solve (e04wdc) and nag_opt_lsq_covariance (e04ycc).