NAG Library Function Document

nag_sparse_nherm_matvec (f11xnc)

1 Purpose

nag_sparse_nherm_matvec (f11xnc) computes a matrix-vector or conjugate transposed matrix-vector product involving a complex sparse non-Hermitian matrix stored in coordinate storage format.

2 Specification

```
#include <nag.h>
#include <nagf11.h>
```

```
void nag_sparse_nherm_matvec (Nag_TransType trans, Integer n, Integer nnz,
     const Complex a[], const Integer irow[], const Integer icol[],
     Nag_SparseNsym_CheckData check, const Complex x[], Complex y[],
     NagError *fail)
```

3 Description

nag_sparse_nherm_matvec (f11xnc) computes either the matrix-vector product $y = Ax$, or the conjugate transposed matrix-vector product $y = A^{\mathrm{H}}x$, according to the value of the argument **trans**, where A is a complex n by n sparse non-Hermitian matrix, of arbitrary sparsity pattern. The matrix A is stored in coordinate storage (CS) format (see Section 2.1.1 in the f11 Chapter Introduction). The array **a** stores all the nonzero elements of A, while arrays **irow** and **icol** store the corresponding row and column indices respectively.

It is envisaged that a common use of nag_sparse_nherm_matvec (f11xnc) will be to compute the matrix-vector product required in the application of nag_sparse_nherm_basic_solver (f11bsc) to sparse complex linear systems. This is illustrated in Section 10 in nag_sparse_nherm_precon_ssor_solve (f11drc).

4 References

None.

5 Arguments

1: **trans** – Nag_TransType *Input*

On entry: specifies whether or not the matrix A is conjugate transposed.

trans = Nag_NoTrans
$y = Ax$ is computed.

trans = Nag_ConjTrans
$y = A^{\mathrm{H}}x$ is computed.

Constraint: **trans** = Nag_NoTrans or Nag_ConjTrans.

2: **n** – Integer *Input*

On entry: n, the order of the matrix A.

Constraint: **n** ≥ 1.

3: **nnz** – Integer *Input*

On entry: the number of nonzero elements in the matrix A.

Constraint: $1 \leq \mathbf{nnz} \leq \mathbf{n}^2$.

4: **a**[**nnz**] – const Complex *Input*

On entry: the nonzero elements in the matrix A, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The function nag_sparse_nherm_sort (f11znc) may be used to order the elements in this way.

5: **irow**[**nnz**] – const Integer *Input*
6: **icol**[**nnz**] – const Integer *Input*

On entry: the row and column indices of the nonzero elements supplied in array **a**.

Constraints:

$1 \leq \mathbf{irow}[i] \leq \mathbf{n}$ and $1 \leq \mathbf{icol}[i] \leq \mathbf{n}$, for $i = 0, 1, \ldots, \mathbf{nnz} - 1$;
$\mathbf{irow}[i-1] < \mathbf{irow}[i]$ or $\mathbf{irow}[i-1] = \mathbf{irow}[i]$ and $\mathbf{icol}[i-1] < \mathbf{icol}[i]$, for $i = 1, 2, \ldots, \mathbf{nnz} - 1$.

7: **check** – Nag_SparseNsym_CheckData *Input*

On entry: specifies whether or not the CS representation of the matrix A, values of **n**, **nnz**, **irow** and **icol** should be checked.

check = Nag_SparseNsym_Check
Checks are carried on the values of **n**, **nnz**, **irow** and **icol**.

check = Nag_SparseNsym_NoCheck
None of these checks are carried out.

See also Section 9.2.

Constraint: **check** = Nag_SparseNsym_Check or Nag_SparseNsym_NoCheck.

8: **x**[**n**] – const Complex *Input*

On entry: the vector x.

9: **y**[**n**] – Complex *Output*

On exit: the vector y.

10: **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_INT

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 1.

On entry, **nnz** $= \langle value \rangle$.
Constraint: **nnz** ≥ 1.

NE_INT_2

On entry, **nnz** $= \langle value \rangle$ and **n** $= \langle value \rangle$.
Constraint: **nnz** $\leq$ **n**^{2}.

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_INVALID_CS

On entry, $i = \langle value \rangle$, **icol**$[i-1] = \langle value \rangle$ and **n** $= \langle value \rangle$.
Constraint: **icol**$[i-1] \geq 1$ and **icol**$[i-1] \leq$ **n**.

On entry, $i = \langle value \rangle$, **irow**$[i-1] = \langle value \rangle$ and **n** $= \langle value \rangle$.
Constraint: **irow**$[i-1] \geq 1$ and **irow**$[i-1] \leq$ **n**.

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.

NE_NOT_STRICTLY_INCREASING

On entry, **a**$[i-1]$ is out of order: $i = \langle value \rangle$.

On entry, the location (**irow**$[I-1]$, **icol**$[I-1]$) is a duplicate: $I = \langle value \rangle$. Consider calling nag_sparse_nherm_sort (f11znc) to reorder and sum or remove duplicates.

7 Accuracy

The computed vector y satisfies the error bound:

$$\|y - Ax\|_\infty \leq c(n)\epsilon\|A\|_\infty\|x\|_\infty, \text{ if } \textbf{trans} = \text{Nag\_NoTrans, or}$$

$$\|y - A^{\mathrm{H}}x\|_\infty \leq c(n)\epsilon\|A^{\mathrm{H}}\|_\infty\|x\|_\infty, \text{ if } \textbf{trans} = \text{Nag\_ConjTrans,}$$

where $c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***.

8 Parallelism and Performance

nag_sparse_nherm_matvec (f11xnc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_sparse_nherm_matvec (f11xnc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

9.1 Timing

The time taken for a call to nag_sparse_nherm_matvec (f11xnc) is proportional to **nnz**.

9.2 Use of check

It is expected that a common use of nag_sparse_nherm_matvec (f11xnc) will be to compute the matrix-vector product required in the application of nag_sparse_nherm_basic_solver (f11bsc) to sparse complex linear systems. In this situation nag_sparse_nherm_matvec (f11xnc) is likely to be called many times with the same matrix A. In the interests of both reliability and efficiency you are recommended to set **check** = Nag_SparseNsym_Check for the first of such calls, and to set **check** = Nag_SparseNsym_NoCheck for all subsequent calls.

10 Example

This example reads in a complex sparse matrix A and a vector x. It then calls nag_sparse_nherm_matvec (f11xnc) to compute the matrix-vector product $y = Ax$ and the conjugate transposed matrix-vector product $y = A^{\mathrm{H}}x$.

10.1 Program Text

```
/* nag_sparse_nherm_matvec (f11xnc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <nag.h>
#include <nag_stdlib.h>
#include <nagf11.h>

int main(void)
{
  /* Scalars */
  Integer exit_status = 0;
  Integer i, j, n, nnz;
  /* Arrays */
  char nag_enum_arg[40];
  Integer *irow = 0, *icol = 0;
  Complex *a = 0, *x = 0, *y = 0;
  /* NAG types */
  NagError fail;
  Nag_TransType trans;
  Nag_SparseNsym_CheckData check;

  INIT_FAIL(fail);

  printf("nag_sparse_nherm_matvec (f11xnc) Example Program Results\n");

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

  /* Read order of matrix and number of nonzero entries */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n]", &n);
#else
  scanf("%" NAG_IFMT "%*[^\n]", &n);
#endif
```

```
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n]", &nnz);
#else
  scanf("%" NAG_IFMT "%*[^\n]", &nnz);
#endif

  /* Allocate memory */
  if (!(a = NAG_ALLOC(nnz, Complex)) ||
      !(x = NAG_ALLOC(n, Complex)) ||
      !(y = NAG_ALLOC(n, Complex)) ||
      !(icol = NAG_ALLOC(nnz, Integer)) || !(irow = NAG_ALLOC(nnz, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Read the matrix A */
  for (i = 0; i < nnz; i++)
#ifdef _WIN32
    scanf_s(" ( %lf , %lf ) %" NAG_IFMT "%" NAG_IFMT "%*[^\n]", &a[i].re,
            &a[i].im, &irow[i], &icol[i]);
#else
    scanf(" ( %lf , %lf ) %" NAG_IFMT "%" NAG_IFMT "%*[^\n]", &a[i].re,
          &a[i].im, &irow[i], &icol[i]);
#endif

  /* Read the vector x */
#ifdef _WIN32
  for (j = 0; j < n; j++)
    scanf_s(" ( %lf , %lf )", &x[j].re, &x[j].im);
#else
  for (j = 0; j < n; j++)
    scanf(" ( %lf , %lf )", &x[j].re, &x[j].im);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* Calculate matrix-vector product */
  /* Nag_NoTrans  */
#ifdef _WIN32
  scanf_s("%39s%*[^\n]", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s%*[^\n]", nag_enum_arg);
#endif
  trans = (Nag_TransType) nag_enum_name_to_value(nag_enum_arg);

  /* Nag_SparseNsym_Check  */
#ifdef _WIN32
  scanf_s("%39s%*[^\n]", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s%*[^\n]", nag_enum_arg);
#endif
  check = (Nag_SparseNsym_CheckData) nag_enum_name_to_value(nag_enum_arg);

  /* nag_sparse_nherm_matvec (f11xnc)
   * Complex sparse non-Hermitian matrix vector multiply.
   */
  nag_sparse_nherm_matvec(trans, n, nnz, a, irow, icol, check, x, y, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_sparse_nherm_matvec (f11xnc)\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Output results */
  printf("\n Matrix-vector product\n");
  for (j = 0; j < n; j++)
    printf(" (%13.4e, %13.4e)\n", y[j].re, y[j].im);

  /* Calculate conjugate transposed matrix-vector product */
  /* Nag_ConjTrans  */
#ifdef _WIN32
  scanf_s("%39s%*[^\n]", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s%*[^\n]", nag_enum_arg);
#endif
  trans = (Nag_TransType) nag_enum_name_to_value(nag_enum_arg);

  /* Nag_SparseNsym_NoCheck  */
#ifdef _WIN32
  scanf_s("%39s%*[^\n]", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s%*[^\n]", nag_enum_arg);
#endif
  check = (Nag_SparseNsym_CheckData) nag_enum_name_to_value(nag_enum_arg);

  /* nag_sparse_nherm_matvec (f11xnc)
   * Complex sparse non-Hermitian matrix vector multiply.
   */
  nag_sparse_nherm_matvec(trans, n, nnz, a, irow, icol, check, x, y, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_sparse_nherm_matvec (f11xnc)\n%s\n", fail.message);
    exit_status = 2;
    goto END;
  }

  /* Output results */
  printf("\n Conjugate transposed matrix-vector product\n");
  for (j = 0; j < n; j++)
    printf(" (%13.4e, %13.4e)\n", y[j].re, y[j].im);

END:
  NAG_FREE(a);
  NAG_FREE(x);
  NAG_FREE(y);
  NAG_FREE(icol);
  NAG_FREE(irow);

  return exit_status;
}
```

10.2 Program Data

```
nag_sparse_nherm_matvec (f11xnc) Example Program Data
  5                       : n
 11                       : nnz
( 2., 3.)   1    1
( 1.,-4.)   1    2
( 1., 0.)   2    3
(-1.,-2.)   2    4
( 4., 1.)   3    1
( 0., 1.)   3    3
( 1., 3.)   3    5
( 0.,-1.)   4    4
( 2.,-6.)   4    5
(-2., 0.)   5    2
( 3., 1.)   5    5       : (a, irow, icol)[i], i=0,...,nnz-1
( 0.70, 0.21)
( 0.16,-0.43)
( 0.52, 0.97)
( 0.77, 0.00)
( 0.28,-0.64)             : x[i], i=0,...,n-1
 Nag_NoTrans              : trans
 Nag_SparseNsym_Check     : check
 Nag_ConjTrans            : trans
 Nag_SparseNsym_NoCheck   : check
```

10.3 Program Results

```
nag_sparse_nherm_matvec (f11xnc) Example Program Results

 Matrix-vector product
 (  -7.9000e-01,    1.4500e+00)
 (  -2.5000e-01,   -5.7000e-01)
 (   3.8200e+00,    2.2600e+00)
 (  -3.2800e+00,   -3.7300e+00)
 (   1.1600e+00,   -7.8000e-01)

 Conjugate transposed matrix-vector product
 (   5.0800e+00,    1.6800e+00)
 (  -7.0000e-01,    4.2900e+00)
 (   1.1300e+00,   -9.5000e-01)
 (   7.0000e-01,    1.5200e+00)
 (   5.1700e+00,    1.8300e+00)
```