NAG Library Function Document
nag_sparse_nsym_matvec (f11xac)

1 Purpose

nag_sparse_nsym_matvec (f11xac) computes a matrix-vector or transposed matrix-vector product involving a real sparse nonsymmetric matrix stored in coordinate storage format.

2 Specification

```
#include <nag.h>
#include <nagf11.h>
void nag_sparse_nsym_matvec (Nag_TransType trans, Integer n, Integer nnz,
const double a[], const Integer irow[], const Integer icol[],
Nag_SparseNsym_CheckData check, const double x[], double y[],
NagError *fail)
```

3 Description

nag_sparse_nsym_matvec (f11xac) computes either the matrix-vector product $y = Ax$, or the transposed matrix-vector product $y = A^T x$, according to the value of the argument `trans`, where $A$ is an $n$ by $n$ sparse nonsymmetric 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 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_nsym_matvec (f11xac) will be to compute the matrix-vector product required in the application of nag_sparse_nsym_basic_solver (f11bec) to sparse linear systems. An illustration of this usage appears in Section 10 in nag_sparse_nsym_precon_ssor_solve (f11ddc).

4 References

None.

5 Arguments

1. `trans` – Nag_TransType
   
   *Input*
   
   On entry: specifies whether or not the matrix $A$ is transposed.

   `trans = Nag_NoTrans`
   
   $y = Ax$ is computed.

   `trans = Nag_Trans`
   
   $y = A^T x$ is computed.

   *Constraint:* `trans = Nag_NoTrans` or `Nag_Trans`.

2. `n` – Integer
   
   *Input*
   
   On entry: $n$, the order of the matrix $A$.

   *Constraint:* $n \geq 1$. 
3: \textbf{nnz} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the number of nonzero elements in the matrix \(A\).

\textit{Constraint:} \(1 \leq \text{nnz} \leq n^2\).

4: \textbf{a[nnz]} – const double \hspace{1cm} \textit{Input}

\textit{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 \texttt{nag_sparse_nsym_sort} (f11zac) may be used to order the elements in this way.

5: \textbf{irow[nnz]} – const Integer \hspace{1cm} \textit{Input}

6: \textbf{icol[nnz]} – const Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the row and column indices of the nonzero elements supplied in array \texttt{a}.

\textit{Constraints:}

\texttt{irow} and \texttt{icol} must satisfy the following constraints (which may be imposed by a call to \texttt{nag_sparse_nsym_sort} (f11zac)):

\begin{align*}
1 \leq \text{row}[i] &\leq n \text{ and } 1 \leq \text{icol}[i] \leq n, \text{ for } i = 0, 1, \ldots, \text{nnz} - 1; \\
\text{row}[i-1] &< \text{row}[i] \text{ or } \text{row}[i-1] = \text{row}[i] \text{ and } \text{icol}[i-1] < \text{icol}[i], \text{ for } i = 1, 2, \ldots, \text{nnz} - 1.
\end{align*}

7: \textbf{check} – \texttt{Nag_SparseNsym\_CheckData} \hspace{1cm} \textit{Input}

\textit{On entry:} specifies whether or not the CS representation of the matrix \(A\), values of \texttt{n}, \texttt{nnz}, \texttt{irow} and \texttt{icol} should be checked.

\texttt{check} = \texttt{Nag_SparseNsym\_Check}

Checks are carried on the values of \texttt{n}, \texttt{nnz}, \texttt{irow} and \texttt{icol}.

\texttt{check} = \texttt{Nag_SparseNsym\_NoCheck}

None of these checks are carried out.

See also Section 9.2.

\textit{Constraint:} \texttt{check} = \texttt{Nag_SparseNsym\_Check} or \texttt{Nag_SparseNsym\_NoCheck}.

8: \textbf{x[n]} – const double \hspace{1cm} \textit{Input}

\textit{On entry:} the vector \(x\).

9: \textbf{y[n]} – double \hspace{1cm} \textit{Output}

\textit{On exit:} the vector \(y\).

10: \textbf{fail} – \texttt{NagError \*} \hspace{1cm} \textit{Input/Output}

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

6 \hspace{1cm} \textbf{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 \langle value \rangle had an illegal value.
NE_INT
On entry, $n = \langle \text{value} \rangle$.
Constraint: $n \geq 1$.

On entry, $nnz = \langle \text{value} \rangle$.
Constraint: $nnz \geq 1$.

NE_INT_2
On entry, $nnz = \langle \text{value} \rangle$ and $n = \langle \text{value} \rangle$.
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 3.6.6 in the Essential Introduction for further information.

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

On entry, $i = \langle \text{value} \rangle$, $irow[i-1] = \langle \text{value} \rangle$ and $n = \langle \text{value} \rangle$.
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 3.6.5 in the Essential Introduction for further information.

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

On entry, the location $(irow[I-1],icol[I-1])$ is a duplicate: $I = \langle \text{value} \rangle$. Consider calling
nag_sparse_nsym_sort (f11zac) to reorder and sum or remove duplicates.

7 Accuracy
The computed vector $y$ satisfies the error bound:

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

$\|y - A^Tx\|_\infty \leq c(n)\epsilon\|A^T\|_\infty\|x\|_\infty$, if $\text{trans} = \text{Nag\_Trans}$,

where $c(n)$ is a modest linear function of $n$, and $\epsilon$ is the \textit{machine precision}.

8 Parallelism and Performance
nag_sparse_nsym_matvec (f11xac) is threaded by NAG for parallel execution in multithreaded
implementations of the NAG Library.

nag_sparse_nsym_matvec (f11xac) 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_nsym_matvec (f11xac) is proportional to \( nnz \).

9.2 Use of check

It is expected that a common use of nag_sparse_nsym_matvec (f11xac) will be to compute the matrix-vector product required in the application of nag_sparse_nsym_basic_solver (f11bec) to sparse linear systems. In this situation nag_sparse_nsym_matvec (f11xac) 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 \[ \text{check} = \text{NagSparseNsym_Check} \] for the first of such calls, and to set \[ \text{check} = \text{NagSparseNsym_NoCheck} \] for all subsequent calls.

10 Example

This example reads in a sparse matrix \( A \) and a vector \( x \). It then calls nag_sparse_nsym_matvec (f11xac) to compute the matrix-vector product \( y = Ax \) and the transposed matrix-vector product \( y = A^T x \).

10.1 Program Text

```c
/* nag_sparse_nsym_matvec (f11xac) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011.
*/
#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;
    double *a = 0, *x = 0, *y = 0;
    /* NAG types */
    NagError fail;
    Nag_TransType trans;
    Nag_SparseNsym_CheckData check;
    INIT_FAIL(fail);

    printf("nag_sparse_nsym_matvec (f11xac) 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 non-zero 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_FMT"%[\n]", &nnz);
#endif

/* Allocate memory */
if ( !(a = NAG_ALLOC(nnz, double)) ||
    !(x = NAG_ALLOC(n, double)) ||
    !(y = NAG_ALLOC(n, double)) ||
    !(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"%"NAG_FMT"%"NAG_FMT"%[\n]", &a[i], &irow[i], &icol[i]);
#else
    scanf("%lf"%"NAG_FMT"%"NAG_FMT"%[\n]", &a[i], &irow[i], &icol[i]);
#endif

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

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

#ifdef _WIN32
    scanf_s("%39s%[\n]", nag_enum_arg, _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_nsym_matvec (f11xac) */
* Calculate matrix-vector product without transposed matrix.
*/
    nag_sparse_nsym_matvec(trans, n, nnz, a, irow, icol, check, x, y, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_sparse_nsym_matvec (f11xac)\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("%16.4e\n", y[j]);

/* Calculate transposed matrix-vector product */
#ifdef _WIN32
    scanf_s("%39s%[\n]", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s%[\n]", nag_enum_arg);
#endif
    trans = (Nag_TransType) nag_enum_name_to_value (nag_enum_arg);
```c
#include <nag.h>
#include <nag_enum.h>
#include <nag_error.h>

#define _WIN32
#define _countof(x) (sizeof(x)/sizeof(x[0]))

#include <stdio.h>
#include <stdlib.h>

int main() {
    int n = 5;
    int nnz = 11;
    double a[nnz] = {0.7, 0.16, 0.52, 0.77, 0.28, -0.2, 0.7, 0.52, 0.77, 0.28, -0.2};
    int irow[nnz] = {0, 0, 0, 1, 1, 2, 3, 4, 5, 5, 5};
    int icol[nnz] = {1, 2, 2, 4, 3, 3, 4, 4, 5, 5, 5};
    double x[n] = {1.0, 1.0, 1.0, 1.0, 1.0};
    double y[n];
    Nag_SparseNsym check = Nag_SparseNsym_NoCheck;
    Nag_Trans trans = Nag_Trans;
    Nag_NoTrans trans = Nag_NoTrans;
    Nag_SparseNsym_Check check = Nag_SparseNsym_Check;

    Nag_SparseNsym_CheckData check_data = nag_enum_name_to_value(nag_enum_arg);
    if (check != check_data) {
        printf("Error from nag_sparse_nsym_matvec (f11xac)\n", fail.message);
        exit_status = 2;
        return 0;
    }

    printf("Transposed matrix-vector product\n");
    for (int j = 0; j < n; j++)
        printf("%16.4e\n", y[j]);
}
```

```
10.2 Program Data

nag_sparse_nsym_matvec (f11xac) Example Program Data

<table>
<thead>
<tr>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nnz</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a, irow, icol[i], i=0,...,nnz-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.70</td>
</tr>
<tr>
<td>0.16</td>
</tr>
<tr>
<td>0.52</td>
</tr>
<tr>
<td>0.77</td>
</tr>
<tr>
<td>0.28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x[i], i=0,...,n-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.70</td>
</tr>
<tr>
<td>0.16</td>
</tr>
<tr>
<td>0.52</td>
</tr>
<tr>
<td>0.77</td>
</tr>
<tr>
<td>0.28</td>
</tr>
</tbody>
</table>

Nag_NoTrans : trans
Nag_SparseNsym_Check : check
Nag_Trans : trans
Nag_SparseNsym_NoCheck : check
```
10.3 Program Results

nag_sparse_nsym_matvec (f11xac) Example Program Results

Matrix-vector product
  1.5600e+00
  -2.5000e-01
  3.6000e+00
  1.3300e+00
  5.2000e-01

Transposed matrix-vector product
  3.4800e+00
  1.4000e-01
  6.8000e-01
  6.1000e-01
  2.9000e+00