1 Purpose

nag_sparse_herm_precon_ssor_solve (f11jrc) solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a complex sparse Hermitian matrix, represented in symmetric coordinate storage format.

2 Specification

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

void nag_sparse_herm_precon_ssor_solve (Integer n, Integer nnz,
   const Complex a[], const Integer irow[], const Integer icol[],
   const double rdiag[], double omega, Nag_SparseSym_CheckData check,
   const Complex y[], Complex x[], NagError *fail)
```

3 Description

nag_sparse_herm_precon_ssor_solve (f11jrc) solves a system of equations

\[ Mx = y \]

involving the preconditioning matrix

\[ M = \frac{1}{\omega(2 - \omega)}(D + \omega L)(D^{-1} + \omega L)^H \]

corresponding to symmetric successive-over-relaxation (SSOR) (see Young (1971)) on a linear system \( Ax = b \), where \( A \) is a sparse complex Hermitian matrix stored in symmetric coordinate storage (SCS) format (see Section 2.1.2 in the f11 Chapter Introduction).

In the definition of \( M \) given above \( D \) is the diagonal part of \( A \), \( L \) is the strictly lower triangular part of \( A \) and \( \omega \) is a user-defined relaxation parameter. Note that since \( A \) is Hermitian the matrix \( D \) is necessarily real.

4 References


5 Arguments

1: \( n \) – Integer \( \text{Input} \)

\( On \ entry: n \), the order of the matrix \( A \).

\( Constraint: n \geq 1. \)

2: \( nnz \) – Integer \( \text{Input} \)

\( On \ entry: \) the number of nonzero elements in the lower triangular part of the matrix \( A \).

\( Constraint: 1 \leq nnz \leq n \times (n + 1)/2. \)
On entry: the nonzero elements in the lower triangular part of 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_herm_sort (f11zpc) may be used to order the elements in this way.

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

Constraints:

$\text{irow}$ and $\text{icol}$ must satisfy the following constraints (which may be imposed by a call to nag_sparse_herm_sort (f11zpc)):

\[
1 \leq \text{irow}[i] \leq n \quad \text{and} \quad 1 \leq \text{icol}[i] \leq \text{irow}[i], \quad \text{for} \quad i = 0, 1, \ldots, \text{nnz} - 1;
\]

$\text{irow}[i - 1] < \text{irow}[i]$ or $\text{irow}[i - 1] = \text{irow}[i]$ and $\text{icol}[i - 1] < \text{icol}[i]$, for

$i = 1, 2, \ldots, \text{nnz} - 1$.

On entry: the elements of the diagonal matrix $D^{-1}$, where $D$ is the diagonal part of $A$. Note that since $A$ is Hermitian the elements of $D^{-1}$ are necessarily real.

On entry: the relaxation parameter $\omega$.

Constraint: $0.0 < \omega < 2.0$.

On entry: specifies whether or not the input data should be checked.

Checks are carried out on the values of $n$, nnz, irow, icol and omega.

None of these checks are carried out.

Constraint: check = Nag_SparseSym_Check or Nag_SparseSym_NoCheck.

On entry: the right-hand side vector $y$.

On exit: the solution vector $x$.

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument $\langle value\rangle$ had an illegal value.
On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 1 \).

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

On entry, \( nnz = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( nnz \leq n \times (n + 1)/2 \).

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.

On entry, \( I = \langle \text{value} \rangle \), \( irow[I - 1] = \langle \text{value} \rangle \), \( icol[I - 1] = \langle \text{value} \rangle \).
Constraint: \( 1 \leq icol[i - 1] \leq irow[i - 1] \).

On entry, \( I = \langle \text{value} \rangle \), \( irow[I - 1] = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( 1 \leq irow[i - 1] \leq n \).

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.

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_herm_sort (f11zpc) to reorder and sum or remove duplicates.

On entry, \( \omega = \langle \text{value} \rangle \).
Constraint: \( 0 < \omega < 2.0 \).

The matrix \( A \) has no diagonal entry in row \( \langle \text{value} \rangle \).

The computed solution \( x \) is the exact solution of a perturbed system of equations \((M + \delta M)x = y\), where

\[
|\delta M| \leq c(n)\epsilon\|D + \omega L\|D^{-1}\|(D + \omega L)^T|,
\]

\( c(n) \) is a modest linear function of \( n \), and \( \epsilon \) is the machine precision.

Parallelism and Performance
Not applicable.
9 Further Comments

9.1 Timing

The time taken for a call to nag_sparse_herm_precon_ssor_solve (f11jrc) is proportional to \( \text{nnz} \).

10 Example

This example program solves the preconditioning equation \( Mx = y \) for a 9 by 9 sparse complex Hermitian matrix \( A \), given in symmetric coordinate storage (SCS) format.

10.1 Program Text

```c
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf11.h>

int main(void)
{
    /* Scalars */
    Integer exit_status = 0;
    double omega;
    Integer i, n, nnz;
    /* Arrays */
    char nag_enum_arg[100];
    Complex *a = 0, *x = 0, *y = 0;
    double *rdiag = 0;
    Integer *icol = 0, *irow = 0;
    /* NAG types */
    Nag_SparseSym_CheckData check;
    NagError fail;

    INIT_FAIL(fail);
    printf("nag_sparse_herm_precon_ssor_solve (f11jrc) Example Program Results");
    printf("\n");
    /* Skip heading in data file*/
    #ifdef _WIN32
        scanf_s("%*[\n"]);
    #else
        scanf("%*[\n"]);
    #endif
    /* Read algorithmic parameters*/
    #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)) ||
        !(rdiag = NAG_ALLOC(n, double)) ||
```
!((icol = NAG_ALLOC(nnz, Integer)) ||
!(irow = NAG_ALLOC(nnz, Integer))
)
{
printf("Allocation failure\n");
exit_status = -1;
go to END;
}
#endif
#define _WIN32
scanf_s("%99s%\n", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%99s%\n", nag_enum_arg);
#endif
/* nag_enum_name_to_value (x04nac).
* Converts NAG enum member name to value *
*/
check = (Nag_SparseSym_CheckData) nag_enum_name_to_value(nag_enum_arg);
#endif
#define _WIN32
scanf_s("%lf%\n", &omega);
#else
scanf("%lf%\n", &omega);
#endif
/* Read the matrix a */
for (i = 0; i < nnz; i++)
#endif
#define _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 rhs vector y */
for (i = 0; i < n; i++)
#endif
#define _WIN32
scanf_s(" ( %lf , %lf ) ", &y[i].re, &y[i].im);
#else
scanf(" ( %lf , %lf ) ", &y[i].re, &y[i].im);
#endif
#endif
#define _WIN32
scanf("%\n");
#else
scanf("%\n");
#endif
/* Fill in the diagonal part */
for (i = 0; i < nnz; i++)
if (irow[i] == icol[i])
    rdiag[irow[i]-1] = 1.0/(double)(a[i].re);

/* nag_sparse_herm_precon_ssor_solve (f11jrc).
* Solution of linear system involving preconditioning matrix
* generated by applying SSOR to complex sparse Hermitian matrix
*/
nag_sparse_herm_precon_ssor_solve(n, nnz, a, irow, icol, rdiag, omega, check, y, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_sparse_herm_precon_ssor_solve (f11jrc)\n\n", fail.message);
    exit_status = 1;
go to END;
}
/* Output x*/
printf(" Converged Solution\n");
for (i = 0; i < n; i++)
    printf(" (%13.4e, %13.4e)\n", x[i].re, x[i].im);
END:
NAG_FREE(a);
NAG_FREE(x);
NAG_FREE(y);
NAG_FREE(rdiag);
NAG_FREE(icol);
NAG_FREE(irow);
return exit_status;
}

10.2 Program Data

nag_sparse_herm_precon_ssor_solve (f11jrc) Example Program Data

9 : n
23 : nnz
Nag_SparseSym_Check : check
1.1 : omega
( 6., 0.) 1 1
(-1., 1.) 2 1
( 6., 0.) 2 2
( 0., 1.) 3 2
( 5., 0.) 3 3
( 5., 0.) 4 4
( 2.,-2.) 5 1
( 4., 0.) 5 5
( 1., 1.) 6 3
( 2., 0.) 6 4
( 6., 0.) 6 6
(-4., 3.) 7 2
( 0., 1.) 7 5
(-1., 0.) 7 6
( 6., 0.) 7 7
(-1.,-1.) 8 4
( 0.,-1.) 8 6
( 9., 0.) 8 8
( 1., 3.) 9 1
( 1., 2.) 9 5
(-1., 0.) 9 6
( 1., 4.) 9 8
( 9., 0.) 9 9 : a[i], irow[i], icol[i], i=0,...,nnz-1
( 8., 54.)
(-10.,-92.)
(25., 27.)
(26.,-28.)
(54., 12.)
(26.,-22.)
(47., 65.)
(71.,-57.)
(60., 70.) : y[i], i=0,...,n-1

10.3 Program Results

nag_sparse_herm_precon_ssor_solve (f11jrc) Example Program Results

Converged Solution
( 1.0977e+00,  5.9139e+00)
( 2.2304e-01, -1.4085e+01)
( 2.2315e+00,  7.0868e+00)
( 4.8164e+00, -6.1807e+00)
( 6.7632e+00,  1.5690e+00)
( 3.3531e+00, -4.7849e+00)
( 6.6991e-01, -1.4646e+00)
( 8.8315e+00, -3.6326e+00)
( 4.7685e+00,  1.2130e-01)