DOE/ER/25048--T3

DE90 013970

An Overview of NSPCG: A Nonsymmetric Preconditioned Conjugate Gradient Package

by

D. R. Kincaid, T. C. Oppe, and W. D. Joubert

October 1988 CNA-228

An Overview of NSPCG: A Nonsymmetric Preconditioned Conjugate Gradient Package.

David R. Kincaid
Thomas C. Oppe
Wayne D. Joubert

Center for Numerical Analysis
The University of Texas at Austin
Austin, Texas 78713-8510 USA

October 3, 1988

Abstract: The most recent research-oriented software package developed as part of the ITPACK Project is called "NSPCG" since it contains many nonsymmetric preconditioned conjugate gradient procedures. It is designed to solve large sparse systems of linear algebraic equations by a variety of different iterative methods. The coefficient matrix can be passed in one of several different matrix data storage schemes. These sparse data formats allow matrices with a wide range of structures from highly structured ones such as those with all nonzeros along a relatively small number of diagonals to completely unstructured sparse matrices. Alternatively, the package allows the user to call the accelerators directly with user-supplied routines for performing certain matrix operations. In this case, one can use the data format from an application program and not be required to copy the matrix into one of the package formats. This is particularly advantageous when memory space is limited.

The main entry point into the package is through a single subroutine call. The various methods are accessed by using a particular naming convention for the first two parameters that in turn selects a preconditioner, an accelerator, and a data storage scheme. Some of the basic preconditioners that are available are Jacobi, Incomplete LU Decomposition, and Symmetric Successive Overrelaxation as well as block preconditioners. The user can select from a large collection of accelerators such as Conjugate Gradient (CG), Chebyshev (SI, for semi-iterative), Generalized Minimal Residual (GMRES), Biconjugate Gradient Squared (BCGS), and many others. The package is modular so that almost any accelerator can be used with almost any preconditioner. One of the main purposes for the development of the package was to provide a common modular structure for research on iterative methods. The entire package is written in Fortran 77 with vectorization in mind for applications on supercomputers.

1 Introduction

NSPCG was developed as part of the ITPACK Project at the Center for Numerical Analysis. The ITPACK software packages are designed to solve $Au = b$ by various iterative techniques where A is a large, sparse, real matrix. The following packages have been developed:

ITPACK 1: This is a prototype package that is no longer available. [See, for example, Kincaid and Young [15], Kincaid and Grimes [14], Kincaid, Grimes, and Young [12], or Grimes, Kincaid, MacGregor and Young [2].]

ITPACK 2C: In this package, A must be symmetric and positive definite (SPD) or nearly so. A is stored in the `A-JA-IA` sparse matrix format. Either conjugate gradient or Chebyshev acceleration is applied to the Jacobi, Symmetric Successive Overrelaxation (SSOR), or Reduced System (RS) basic iterative methods. Successive Overrelaxation (SOR) is supplied without acceleration. Either the natural ordering or a red-black ordering of the unknowns can be used. Adaptive procedures can be used for iteration parameters such as ω and for eigenvalue estimates. [See Kincaid, Respess, Young and Grimes [11].]

ITPACKV 2C: This package is a vectorized version of ITPACK 2C. The iterative algorithms are unchanged, but the storage format for A was changed to allow vectorization of certain computational kernels such as the matrix-vector product, matrix permutation and matrix scaling. A is stored in the `COEF-JCOEF` sparse storage format. This version is available in the ELLPACK package as several solution modules. Versions for the Cyber 205, Cray-1, and Cray X-MP supercomputers have been written. [See Kincaid, Oppe, Respess, Young [7].]

ITPACK 3A: This package contains acceleration routines and basic iterative methods for both symmetric and nonsymmetric systems. The accelerators for symmetric systems are conjugate gradient and Chebyshev acceleration. The accelerators for nonsymmetric systems are ORTHOMIN(s), ORTHODIR(s), ORTHORES(s), Lanczos/ORTHOMIN (Biconjugate Gradient), Lanczos/ORTHODIR, Lanczos/

ORTHORES, and nonsymmetric Chebyshev acceleration. The matrix is stored in the `A-JA-IA` sparse matrix format. [See Young and Mai [24].]

ITPACK 3B: This package contains basically the same algorithms as ITPACK 3A. It is written with an ELLPACK-style preprocessor that allows the user more flexibility in designing the iterative method. The preprocessor constructs a Fortran program with the appropriate calls to the ITPACK 3B library. [See Mai and Young [17].]

NSPCG: (a.k.a. ITPACK 4) This package is similar to ITPACK 3A in that it is Fortran-callable and contains acceleration schemes and basic iterative methods intended for both symmetric and nonsymmetric matrix problems. It has a large variety of basic iterative methods and acceleration schemes, and the matrix can be represented in any one of several sparse matrix formats plus a matrix-free format in which the user supplies the matrix-vector routines. As with ITPACKV 2C, it has been vectorized for vector supercomputers such as the Cyber 205 and Cray X-MP. [See Oppe, Joubert and Kincaid [18].]

The objective of this paper is to give an overview of the most recent research-oriented software package, NSPCG, and to highlight a few of its key features. More detailed information can be found in the User's Guide by Oppe, Joubert, and Kincaid [18].

2 Features of the Package

Some of the key features available in the NSPCG Package are:

- Symmetric accelerators. [conjugate gradient, Chebyshev]
- Nonsymmetric accelerators. [ORTHOMIN, GCR, Lanczos, LSQR, etc.]
- Basic preconditioners. [Jacobi, ILU(k), MILU(k), SOR, SSOR, RS, polynomial, etc.— most have line versions available.]

- Natural, red-black, line red-black, general multicolor orderings of the unknowns and equations allowed. [Corresponding multicolor versions of preconditioners are available.]

- Some accelerators allow left-, right-, or two-sided orientations of the preconditioner.

  Left: $Q^{-1}Au = Q^{-1}b$
  Right: $(AQ^{-1})(Qu) = b$
  Two-Sided: $(Q_L^{-1}AQ_R^{-1})(Q_Ru) = (Q_L^{-1}b)$ assuming $Q = Q_LQ_R$

- Modular Structure. [Any preconditioner can be used with any accelerator and almost any preconditioner with any data storage format.]

- Several sparse matrix storage schemes are available. [The data structures were chosen for efficiency on vector computers and for handling structured or unstructured matrices.]

- Matrix-free mode of usage. [The user supplies customized routines for matrix-vector operations.]

- Wide selection of stopping tests including the idealized stopping test.

- Adaptive procedures for eigenvalue estimates and iteration parameters in the symmetric case. [e.g., ω for the SOR and SSOR methods.]

3 Purposes for Development of the Package

NSPCG was developed for a variety of reasons, among them are the following:

- To investigate the suitability of various basic iterative methods for vector supercomputers.

- To provide a common modular structure for research on iterative methods. [The package is constructed to facilitate the addition of new preconditioners and new acceleration schemes.]

- To serve as an experimental research tool for evaluating iterative methods.

- To guide in the constructing of iterative algorithms tailored to a specific problem. [NSPCG is *not* production software but can provide information useful in designing production iterative code.]

The degree of vectorization depends on many factors including the particular iterative method, the underlying structure of the matrix, the data storage format, the ordering of the equations, and the architecture of the computer. The NSPCG package permits several sparse matrix data structures (suitable for regularly or irregularly structured matrices), various orderings for enhanced vectorization, and different vectorizing philosophies (Cyber 205 memory-to-memory or Cray register-to-register).

NSPCG can provide useful information such as the convergence or nonconvergence of an iterative method, the number of iterations for convergence, the suitability of an approximate stopping test vs. the idealized stopping test, the effectiveness of certain adaptive procedures for iterative parameters, the existence of a preconditioner (e.g., Does ILU or MILU result in negative pivots?), and information on vectorization techniques for various iterative kernels with certain data structures.

Many factors must be considered when designing software for solving linear systems using iterative solution methods. For example, an application may suggest a particular iterative method. Some of the properties of the matrix A that are useful in choosing an iterative method or operator representation are the matrix's sparsity pattern, symmetry in structure or elements, and the existence of constant coefficients. Also, vectorization and parallelization of iterative methods usually depends on exploiting the matrix's sparsity pattern. The representation of A may be either explicit or implicit. For explicit representation, the sparse matrix data structure can be chosen on the basis of storage and computational efficiency. On the other hand, storage requirements might preclude storing A, and the operation Au would have to be represented implicitly.

4 Usage

The calling sequence for the package is

```
CALL NSPCG (⟨precon⟩ , ⟨accel⟩ , NDIM,MDIM,N,MAXNZ,COEF,
  JCOEF,P,IP,U,UBAR,RHS,WKSP,IWKSP,NW,INW,IPARM,RPARM,IER)
```

and a brief summary of the parameters involved is

⟨*precon*⟩	name of preconditioning routine
⟨*accel*⟩	name of acceleration routine
NDIM	row dimension of COEF array
MDIM	column dimension of COEF array
N	order of linear system
MAXNZ	active column size of COEF array
COEF	coefficient matrix (nonzeros) in various formats
JCOEF	integer array associated with COEF
P	permutation vector for multi-color orderings
IP	inverse permutation vector
U	iterative solution vector
UBAR	optional exact solution vector
RHS	right-hand-side vector
WKSP	real workspace vector of length NW
IWKSP	integer workspace vector of length INW
NW	length of WKSP
INW	length of IWKSP
IPARM	integer parameter vector
RPARM	real parameter vector
IER	error flag

5 Choices for Preconditioner ⟨*precon*⟩

The naming convention used for the routine specifying the preconditioner, ⟨*precon*⟩, is ⟨*name*⟩i where ⟨*name*⟩ indicates the preconditioner and i indicates the storage format. The choices for storage format that are available are:

$i = 1$	primary (ELLPACK) format
2	symmetric diagonal format
3	nonsymmetric diagonal format
4	symmetric coordinate format
5	nonsymmetric coordinate format

A brief summary of the choices for preconditioner is given in the following table.

Point Preconditioners	
RICH*i*	Richardson
JAC*i*	Jacobi
SOR*i*	Successive Overrelaxation
SSOR*i*	Symmetric Successive Overrelaxation
IC*i*	Incomplete *LU* decomposition
MIC*i*	Modified Incomplete *LU* decomposition
LSP*i*	Least Squares Polynomial
NEU*i*	Neumann Polynomial
RS*i*	Reduced System
Line or Block Preconditioners	
LJAC*i*	Line Jacobi
LJACX*i*	Line Jacobi (approximate inverse)
LSOR*i*	Line Successive Overrelaxation
LSSOR*i*	Line Symmetric Successive Overrelaxation
BIC*i*	Block Incomplete *LU* decomposition (version 1)
BICX*i*	Block Incomplete *LU* decomposition (version 2)
MBIC*i*	Modified Block Incomplete *LU* decomposition (version 1)
MBICX*i*	Modified Block Incomplete *LU* decomposition (version 2)
LLSP*i*	Line Least Squares Polynomial
LNEU*i*	Line Neumann Polynomial
RS*i*	Reduced System

6 Choices for Accelerator $\langle accel \rangle$

A large collection of accelerators is available to handle both symmetric and nonsymmetric systems. A brief summary of the accelerators in the package is given in the following table.

Symmetric and Positive Definite Case	
CG	conjugate gradient (2-term form)
SI	Chebyshev or Semi-Iteration (2-term form)
SOR	Successive Overrelaxation
SRCG	adaptive Symmetric Successive Overrelaxation CG
SRSI	adaptive Symmetric Successive Overrelaxation SI
Nonsymmetric Case	
BASIC	null accelerator (just basic iterative method)
ME	Minimal Error algorithm (Fridman)
CGNR	conjugate gradient applied to the normal equations (Elman)
LSQR	least squares algorithm (Paige, Saunders)
ODIR	truncated/restarted ORTHODIR (Young, Jea)
OMIN	truncated/restarted ORTHOMIN (Young, Jea)
ORES	truncated/restarted ORTHORES (Young, Jea)
IOM	Incomplete Orthogonalization Method (Saad)
GMRES	Generalized Minimal Residual Method (Saad)
USYMLQ	Unsymmetric LQ (Yip, Saunders, Simon)
USYMQR	Unsymmetric QR (Yip, Saunders, Simon)
LANDIR	Lanczos/ORTHODIR (Young, Jea)
LANMIN	Lanczos/ORTHOMIN or Biconjugate Gradient (Young, Jea)
LANRES	Lanczos/ORTHORES or two-sided Lanczos (Young, Jea)
CGCR	Constrained Generalized Conjugate Residual (Wallis)
BCGS	Biconjugate Gradient Squared (Sonneveld)

7 Storage Modes

NSPCG allows a number of different storage modes for representing the coefficient matrix A using two arrays `COEF` and `JCOEF`. A short description of each storage mode follows.

7.1 Symmetric Diagonal Storage Format

`COEF` Real array of size `N` by `MAXNZ` containing the main and nonzero upper diagonals of A in its columns. Dimensioned `NDIM` by `MDIM` where `NDIM` $\geq$ `N` and `MDIM` $\geq$ `MAXNZ`. `MAXNZ` is the number of diagonals stored. Diagonals are top-justified.

`JCOEF` Integer array of size `MAXNZ` containing nonnegative integers giving the distances of each diagonal from the main diagonal.

For example, the matrix

$$A = \begin{pmatrix} 11 & 12 & 0 & 14 & 0 \\ 12 & 22 & 23 & 0 & 25 \\ 0 & 23 & 33 & 34 & 0 \\ 14 & 0 & 34 & 44 & 45 \\ 0 & 25 & 0 & 45 & 55 \end{pmatrix}$$

would be represented in the `COEF` and `JCOEF` arrays as

$$\texttt{COEF} = \begin{pmatrix} 11 & 12 & 14 \\ 22 & 23 & 25 \\ 33 & 34 & 0 \\ 44 & 45 & 0 \\ 55 & 0 & 0 \end{pmatrix} \qquad \texttt{JCOEF} = \begin{pmatrix} 0 & 1 & 3 \end{pmatrix}$$

This storage mode is intended for diagonal or block structured matrices. On vector computers, most matrix operations vectorize with this format. However, it is the most rigid of the available storage formats and matrix permutation is awkward.

7.2 Nonsymmetric Diagonal Storage Format

`COEF` Real array of size `N` by `MAXNZ` containing the nonzero diagonals of A in its columns. Upper diagonals are top-justified and lower diagonals are bottom-justified.

`JCOEF` Integer array of size `MAXNZ` containing integers giving the distances (positive for upper diagonals and negative for lower diagonals) of each diagonal from the main diagonal.

For example, the matrix

$$A = \begin{pmatrix} 11 & 10 & 0 & 14 & 0 \\ 12 & 22 & 21 & 0 & 25 \\ 0 & 23 & 33 & 32 & 0 \\ 30 & 0 & 34 & 44 & 43 \\ 0 & 25 & 0 & 45 & 55 \end{pmatrix}$$

would be represented in the COEF and JCOEF arrays as

$$\texttt{COEF} = \begin{pmatrix} 11 & 14 & 10 & 0 & 0 \\ 22 & 25 & 21 & 12 & 0 \\ 33 & 0 & 32 & 23 & 0 \\ 44 & 0 & 43 & 34 & 30 \\ 55 & 0 & 0 & 45 & 25 \end{pmatrix} \qquad \texttt{JCOEF} = \begin{pmatrix} 0 & 3 & 1 & -1 & -3 \end{pmatrix}$$

The code for matrix-vector multiplication with this storage scheme is

```
      DO 10 I = 1,N
         Y(I) = 0.0
10    CONTINUE
      DO 20 J = 1,MAXNZ
         NDEL = JCOEF(J)
         IBGN = MAX0 (1,1-NDEL)
         IEND = MIN0 (N,N-NDEL)
         DO 15 I = IBGN,IEND
            Y(I) = Y(I) + COEF(I,J)*X(I+NDEL)
15       CONTINUE
20    CONTINUE
```

7.3 Primary (ELLPACK) Storage Format

COEF Real array of size N by MAXNZ containing all the nonzeros of A. Nonzeros in row i of A appear in row i of COEF. MAXNZ is the maximum number of nonzeros per row. Rows with fewer than MAXNZ nonzeros are padded with zeros in COEF.

JCOEF Integer array of size N by MAXNZ containing the column numbers of corresponding entries in COEF.

For example, the matrix

$$A = \begin{pmatrix} 11 & 0 & 0 & 14 & 15 \\ 0 & 22 & 0 & 0 & 0 \\ 0 & 0 & 33 & 0 & 0 \\ 14 & 0 & 0 & 44 & 45 \\ 15 & 0 & 0 & 45 & 55 \end{pmatrix}$$

would be represented in the COEF and JCOEF arrays as

$$\texttt{COEF} = \begin{pmatrix} 11 & 14 & 15 \\ 22 & 0 & 0 \\ 33 & 0 & 0 \\ 44 & 14 & 45 \\ 55 & 15 & 45 \end{pmatrix} \qquad \texttt{JCOEF} = \begin{pmatrix} 1 & 4 & 5 \\ 2 & 0 & 0 \\ 3 & 0 & 0 \\ 4 & 1 & 5 \\ 5 & 1 & 4 \end{pmatrix}$$

This storage mode is intended for unstructured matrices having a relatively constant number of nonzeros per row. On vector computers certain operations vectorize with this data format while others do not. For example, matrix-vector products, matrix scaling, and matrix permuting all vectorize. Forward and back solution steps, with the natural ordering, do not vectorize in NSPCG with this data format.

The code for matrix-vector multiplication with this storage scheme is

```
      DO 10 I = 1,N
         Y(I) = 0.0
10    CONTINUE
      DO 20 J = 1,MAXNZ
         DO 15 I = 1,N
            Y(I) = Y(I) + COEF(I,J)*X(JCOEF(I,J))
15       CONTINUE
20    CONTINUE
```

7.4 Symmetric Coordinate Storage Format

COEF Real vector of length MAXNZ containing the nonzeros of A in any order. Only the nonzeros on the main diagonal and upper triangle are stored. COEF is dimensioned to be of length NDIM where NDIM $\geq$ MAXNZ.

JCOEF Integer array of size MAXNZ by 2 containing the row numbers of corresponding entries in COEF in column 1 and the column numbers in column 2. Thus if COEF(k) = $a_{i,j}$, then JCOEF(k, 1) = i and JCOEF(k, 2) = j. JCOEF is dimensioned to be of size NDIM by 2.

For example, the matrix

$$A = \begin{pmatrix} 11 & 12 & 0 & 14 & 0 \\ 12 & 22 & 23 & 0 & 25 \\ 0 & 23 & 33 & 34 & 0 \\ 14 & 0 & 34 & 44 & 45 \\ 0 & 25 & 0 & 45 & 55 \end{pmatrix}$$

would be represented in the COEF and JCOEF arrays as

$$\text{COEF} = \begin{pmatrix} 11 \\ 22 \\ 33 \\ 44 \\ 55 \\ 12 \\ 23 \\ 34 \\ 45 \\ 14 \\ 25 \end{pmatrix} \qquad \text{JCOEF} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \\ 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 1 & 4 \\ 2 & 5 \end{pmatrix}$$

This storage mode is the most general of the available storage formats and is intended for unstructured matrices. On vector computers, the matrix-vector product, matrix scaling, and matrix permuting all vectorize with this data format.

7.5 Nonsymmetric Coordinate Storage Format

This format is similar to that of symmetric coordinate storage except that all nonzeros are stored. For example, the matrix

$$A = \begin{pmatrix} 11 & 10 & 0 & 14 & 0 \\ 12 & 22 & 21 & 0 & 25 \\ 0 & 23 & 33 & 32 & 0 \\ 30 & 0 & 34 & 44 & 43 \\ 0 & 25 & 0 & 45 & 55 \end{pmatrix}$$

would be represented in the COEF and JCOEF arrays as

$$\texttt{COEF} = \begin{pmatrix} 11 \\ 22 \\ 33 \\ 44 \\ 55 \\ 14 \\ 25 \\ 10 \\ 21 \\ 32 \\ 43 \\ 12 \\ 23 \\ 34 \\ 45 \\ 30 \\ 25 \end{pmatrix} \qquad \texttt{JCOEF} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \\ 1 & 4 \\ 2 & 5 \\ 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 2 & 1 \\ 3 & 2 \\ 4 & 3 \\ 5 & 4 \\ 4 & 1 \\ 5 & 2 \end{pmatrix}$$

The code for matrix-vector multiplication with this storage scheme is

```
      DO 10 I = 1,N
         Y(I) = 0.0
10    CONTINUE
      DO 15 K = 1,MAXNZ
         Y(JCOEF(K,1)) = Y(JCOEF(K,1)) + COEF(K)*X(JCOEF(K,2))
15    CONTINUE
```

Loop 15 is performed in partitions that have unique JCOEF entries to avoid "smashing" the Y vector during the scatter.

7.6 Using NSPCG in Matrix Format-Free Mode

One feature of the NSPCG package is the ability to call acceleration routines directly with user-supplied customized routines for performing certain matrix operations. The motivation is to allow the user to design an iterative algorithm suitable to a particular application using the data format available in the application code. This is a particularly desirable feature when storage demands may preclude copying the coefficient matrix into the allowable formats in NSPCG. Certain routines for matrix operations must be supplied by the user such as

SUBA	to compute $y = Ax$ given x
SUBAT	to compute $y = A^T x$ given x
SUBQL	to solve $Q_L y = x$ for y given x
SUBQR	to solve $Q_R y = x$ for y given x
SUBQLT	to solve $Q_L^T y = x$ for y given x
SUBQRT	to solve $Q_R^T y = x$ for y given x

Here $Q = Q_L Q_R$ is the splitting matrix, and $Q_L^{-1} A Q_R^{-1}$ is the preconditioned matrix. Note that $Q_L = Q$ and $Q_R = I$ for left preconditioning and $Q_L = I$ and $Q_R = Q$ for right preconditioning. A sample accelerator call might appear as

```
LSQRW (SUBA,SUBAT,SUBQL,SUBQLT,SUBQR,SUBQRT,COEF,JCOEF,
       WFAC,JWFAC,N,U,UBAR,RHS,WKSP,NW,IPARM,RPARM,IER)
```

8 Stopping Tests

A wide selection of stopping tests is available for experimentation. For example, a representative stopping test is

$$\frac{\texttt{EMAX}}{\texttt{EMIN}} \left[\frac{\langle r^{(n)}, \tilde{z}^{(n)} \rangle}{\langle b, Q^{-1} b \rangle} \right]^{\frac{1}{2}} < \zeta \qquad (1)$$

Here, EMAX = RPARM(2) and EMIN = RPARM(3) are estimates of the 2-norm of the preconditioned matrix and its inverse. In the symmetric case, EMAX and EMIN are estimates of the maximum and minimum eigenvalues of $Q^{-1}A$, respectively. Some quantities used in the stopping test are Q, the preconditioning matrix, $r^{(n)} = b - Au^{(n)}$, the current residual, and

$\tilde{z}^{(n)} = Q^{-1}r^{(n)} = Q_R^{-1}Q_L^{-1}r^{(n)}$, the current pseudo-residual. Certain vectors and inner products come for free with certain accelerators. For some accelerators, EMAX and EMIN are adaptively computed. Thus, some stopping tests may be cheap or free.

9 IPARM and RPARM Parameter Arrays

Two arrays of integer and real parameters, IPARM and RPARM, are provided to control certain iteration parameters that affect the performance of the iterative method and are used to communicate with adaptive procedures. For example, some typical values stored in RPARM are

EMAX, EMIN	are eigenvalue estimates of $Q^{-1}A$
OMEGA	is the SOR and SSOR overrelaxation parameter
ALPHAB, BETAB	are SSOR parameters
SPECR	is the spectral radius of the SOR matrix

Default values for all parameters can be set and then a few selected ones can be changed as in the following example.

```
CALL DFAULT (IPARM,RPARM)
IPARM(4) = 3
RPARM(1) = 1.0E-8
```

10 Using Reduced System Methods

The matrix A may have point or block "Property A," in which case the system $Au = b$ can be permuted to the form of a red-black system:

$$\begin{pmatrix} D_R & H \\ K & D_B \end{pmatrix} \begin{pmatrix} u_R \\ u_B \end{pmatrix} = \begin{pmatrix} b_R \\ b_B \end{pmatrix}$$

where D_R and D_B are point or block diagonal matrices. By eliminating u_B, the "reduced system" is formed

$$(D_R - HD_B^{-1}K)u_R = b_R - HD_B^{-1}b_B$$

The NSPCG package has the capability to use the reduced system either explicitly or implicitly as outlined below.

Implicit Use of an RS Method: If A has point or line Property A, NSPCG allows the user to run the point or line RS method with the reduced system implicitly used. First, the routine REDBLK is used to determine if A has Property A, and, if so, the permutation vector P that will permute A to a red-black system is constructed. Then NSPCG is called with an RS preconditioner. The reduced system $D_R - HD_B^{-1}K$ is not explicitly computed; rather, the application of this operator to a vector is accomplished using successive applications of K and H.

```
 CALL REDBLK (NDIM,N,MAXNZ,COEF,JCOEF,P,IP,NSTORE,IWKSP,
A             IER)
 CALL NSPCG (RS6,CG,NDIM,MDIM,N,MAXNZ,COEF,JCOEF,P,IP,U,
A            UBAR,RHS,WKSP,IWKSP,NW,INW,IPARM,RPARM,IER)
```

Explicitly Computing the Reduced System: If A has point Property A, NSPCG allows the user to explicitly compute the reduced system, and apply any of the iterative methods in NSPCG to solve the reduced system. The user calls routine RSNSP with the same calling sequence as NSPCG.

```
 CALL REDBLK (NDIM,N,MAXNZ,COEF,JCOEF,P,IP,NSTORE,IWKSP,
A             IER)
 CALL RSNSP (MIC1,CG,NDIM,MDIM,N,MAXNZ,COEF,JCOEF,P,IP,U,
A            UBAR,RHS,WKSP,IWKSP,NW,INW,IPARM,RPARM,IER)
```

11 Sample Usage

In this section, an example is given of using NSPCG to solve the linear system $Ax = b$ that arises from the discretization of the following partial differential equation:

$$\begin{cases} u_{xx} + 2u_{yy} = 0 & \text{on } S = [0,1] \times [0,1] \\ u = 1 + xy & \text{on boundary of } S \end{cases}$$

Using the standard five-point central difference formula with a mesh size of $h = \frac{1}{11}$, the finite difference stencil at node (i,j) is

$$6u_{i,j} - u_{i-1,j} - u_{i+1,j} - 2u_{i,j+1} - 2u_{i,j-1} = 0$$

The coefficient matrix in the resulting linear system is symmetric and positive definite of order 100 with five nonzero diagonals. Symmetric diagonal storage is used to represent the matrix so that only the main diagonal and the two nonzero super-diagonals need to be stored in arrays COEF and JCOEF:

$$\texttt{COEF} = \begin{pmatrix} 6 & -1 & -2 \\ \vdots & \vdots & \vdots \\ 6 & -1 & -2 \end{pmatrix} \qquad \texttt{JCOEF} = (0, 1, 10) \qquad \texttt{MAXNZ} = 3$$

The iterative method used is the Modified Incomplete Cholesky (MIC(0)) method with conjugate gradient acceleration. Thus, $\langle precon \rangle$ = MIC2 and $\langle accel \rangle$ = CG.

The NSPCG calling program is as follows:

```
      REAL COEF(120,4), RHS(100), U(100), WKSP(600), UBAR(1),
     A     RPARM(30)
      INTEGER JCOEF(4), IWKSP(300), IPARM(30), P(1), IP(1)
      EXTERNAL CG, MIC2
           .
           .
           .

      NDIM = 120
      MDIM = 4
      N = 100
      MAXNZ = 3
      NW = 600
      INW = 300
C
      CALL DFAULT (IPARM,RPARM)
C
C ... NOW, RESET SOME DEFAULT VALUES.
C
      IPARM(2) = 50
      IPARM(3) = 3
      RPARM(1) = 1.0E-8
```

```
C
C ... GENERATE AN INITIAL GUESS FOR U AND CALL NSPCG.
C
      CALL VFILL (N,U,0.0)
C
      CALL NSPCG (MIC2,CG,NDIM,MDIM,N,MAXNZ,COEF,JCOEF,P,IP,U,
     A            UBAR,RHS,WKSP,IWKSP,NW,INW,IPARM,RPARM,IER)

           .
           .
           .

      END
```

The output from NSPCG is given below:

```
     INITIAL ITERATIVE PARAMETERS
     PREPROCESSOR AND PRECONDITIONER PARAMETERS
          IPARM(12) =               2    (NSTORE)
          IPARM(13) =               0    (ISCALE)
          IPARM(14) =               0    (IPERM )
          IPARM(15) =               1    (IFACT )
          IPARM(16) =               0    (LVFILL)
          IPARM(17) =               0    (LTRUNC)
          IPARM(18) =               2    (IPROPA)
          IPARM(19) =              -1    (KBLSZ )
          IPARM(20) =              -1    (NBL2D )
          IPARM(21) =               1    (IFCTV )
          IPARM(22) =               1    (IQLR  )
          IPARM(23) =               2    (ISYMM )
          IPARM(24) =               0    (IELIM )
          IPARM(25) =               1    (NDEG  )
          RPARM(13) =  .00000000E+00     (TIMFAC)
          RPARM(14) =  .00000000E+00     (TIMTOT)
          RPARM(15) =  .35500000E-11     (TOL   )
```

```
     RPARM(16) =   .00000000E+00    (AINF  )

INITIAL ITERATIVE PARAMETERS
GENERAL AND ACCELERATION PARAMETERS
     IPARM( 1) =                2    (NTEST )
     IPARM( 2) =               50    (ITMAX )
     IPARM( 3) =                3    (LEVEL )
     IPARM( 4) =                6    (NOUT  )
     IPARM( 5) =                0    (IDGTS )
     IPARM( 6) =                1    (MAXADP)
     IPARM( 7) =                1    (MINADP)
     IPARM( 8) =                1    (IOMGAD)
     IPARM( 9) =                5    (NS1   )
     IPARM(10) =           100000    (NS2   )
     IPARM(11) =                0    (NS3   )
     RPARM( 1) =  .10000000E-07    (ZETA  )
     RPARM( 2) =  .20000000E+01    (EMAX  )
     RPARM( 3) =  .10000000E+01    (EMIN  )
     RPARM( 4) =  .75000000E+00    (FF    )
     RPARM( 5) =  .75000000E+00    (FFF   )
     RPARM( 6) =  .00000000E+00    (TIMIT )
     RPARM( 7) =  .00000000E+00    (DIGIT1)
     RPARM( 8) =  .00000000E+00    (DIGIT2)
     RPARM( 9) =  .10000000E+01    (OMEGA )
     RPARM(10) =  .00000000E+00    (ALPHAB)
     RPARM(11) =  .25000000E+00    (BETAB )
     RPARM(12) =  .00000000E+00    (SPECR )


CG

    INTERMEDIATE OUTPUT AFTER EACH ITERATION
ITERATION           CONVERGENCE      EMAX .       EMIN
     N       S        TEST

     0       0      .99366E+01   .20000E+01   .10000E+01
```

```
     1        1       .46168E-01    .10010E+01    .10010E+01
     2        2       .57189E-02    .20232E+01    .10002E+01
     3        3       .12255E-02    .24807E+01    .10001E+01
     4        4       .23770E-03    .27522E+01    .10000E+01
     5        5       .49325E-04    .28711E+01    .10000E+01
     6        6       .87776E-05    .29024E+01    .10000E+01
     7        7       .16811E-05    .29071E+01    .10000E+01
     8        8       .42316E-06    .29074E+01    .10000E+01
     9        9       .15339E-06    .29075E+01    .10000E+01
    10       10       .38502E-07    .29075E+01    .10000E+01
    11       11       .71532E-08    .29076E+01    .10000E+01

CG  HAS CONVERGED IN    11 ITERATIONS

    FINAL ITERATIVE PARAMETERS
    GENERAL AND ACCELERATION PARAMETERS
         IPARM( 1) =                2    (NTEST )
         IPARM( 2) =               11    (ITMAX )
         IPARM( 3) =                3    (LEVEL )
         IPARM( 4) =                6    (NOUT  )
         IPARM( 5) =                0    (IDGTS )
         IPARM( 6) =                1    (MAXADP)
         IPARM( 7) =                1    (MINADP)
         IPARM( 8) =                1    (IOMGAD)
         IPARM( 9) =                5    (NS1   )
         IPARM(10) =           100000    (NS2   )
         IPARM(11) =                0    (NS3   )
         RPARM( 1) =  .10000000E-07    (ZETA  )
         RPARM( 2) =  .29076287E+01    (EMAX  )
         RPARM( 3) =  .10000004E+01    (EMIN  )
         RPARM( 4) =  .75000000E+00    (FF    )
         RPARM( 5) =  .75000000E+00    (FFF   )
         RPARM( 6) =  .34800000E+00    (TIMIT )
         RPARM( 7) =  .81454998E+01    (DIGIT1)
         RPARM( 8) =  .78457903E+01    (DIGIT2)
         RPARM( 9) =  .10000000E+01    (OMEGA )
```

```
     RPARM(10) =   .00000000E+00    (ALPHAB)
     RPARM(11) =   .25000000E+00    (BETAB )
     RPARM(12) =   .00000000E+00    (SPECR )

FINAL ITERATIVE PARAMETERS
PREPROCESSOR AND PRECONDITIONER PARAMETERS
     IPARM(12) =                2    (NSTORE)
     IPARM(13) =                0    (ISCALE)
     IPARM(14) =                0    (IPERM )
     IPARM(15) =                1    (IFACT )
     IPARM(16) =                0    (LVFILL)
     IPARM(17) =                0    (LTRUNC)
     IPARM(18) =                1    (IPROPA)
     IPARM(19) =               -1    (KBLSZ )
     IPARM(20) =               -1    (NBL2D )
     IPARM(21) =                1    (IFCTV )
     IPARM(22) =                1    (IQLR  )
     IPARM(23) =                2    (ISYMM )
     IPARM(24) =                0    (IELIM )
     IPARM(25) =                1    (NDEG  )
     RPARM(13) =   .22000000E-01    (TIMFAC)
     RPARM(14) =   .51200000E+00    (TIMTOT)
     RPARM(15) =   .35500000E-11    (TOL   )
     RPARM(16) =   .00000000E+00    (AINF  )
```

12 Distribution of Software

A limited number of copies of the software package NSPCG is available for distribution with the understanding that it is intended as a research tool and may undergo further development. The interested reader should write to the address below for additional information on obtaining the distribution tape of the software and the available documentation.

Center for Numerical Analysis
RLM Hall 13.150
University of Texas at Austin
Austin, TX 78713-8510

A nominal fee is involved to cover handling, mailing charges, etc. Also, reports of difficulties encountered plus comments and suggestions are welcome.

As usual with research-oriented software, The University of Texas at Austin and the Center for Numerical Analysis disclaim all warranties with regard to this software package and its documentation. It should be emphasized that it is preliminary, incomplete, and subject to change.

13 Acknowledgements

This work was supported in part by the National Science Foundation under Grant DCR-8518722, by the Department of Energy under Grant DE-AS05-81ER10954, by Cray Research, Inc., under Grant LRTDTD, and by Sandia National Laboratories, under Grant 06-4298 with The University of Texas at Austin.

References

[1] Grimes, Roger G., David R. Kincaid, and David M. Young [1979] "ITPACK 2.0 User's Guide," Report CNA-150, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas.

[2] Grimes, Roger G., David R. Kincaid, William I. MacGregor, and David M. Young [1978] "ITPACK Report: Adaptive Iterative Algorithms Using Symmetric Sparse Storage," Report CNA-139, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas.

[3] Hageman, Louis A., and David M. Young, [1981] *Applied Iterative Methods*, Academic Press, New York.

[4] Kincaid, David R., Thomas C. Oppe, and David M. Young [1986a] "Vector Computations for Sparse Linear Systems," *SIAM J. Alg. Disc. Math.*, Vol. 7, No. 1, pp. 99-112.

[5] Kincaid, David R., Thomas C. Oppe, and David M. Young [1986b] "Vectorized Iterative Methods for Partial Differential Equations," *Communications in Applied Numerical Methods*, Vol. 2, No. 3, pp. 289-296.

[6] Kincaid, David R., Thomas C. Oppe, John R. Respess, and David M. Young [1985] "Chapter 7: ITPACK Solution Modules," in *Solving Elliptic Problems Using ELLPACK* (John R. Rice and Ronald F. Boisvert, eds.) Springer-Verlag, New York, pp. 237-258.

[7] Kincaid, David R., Thomas C. Oppe, John R. Respess, and David M. Young [1984] "ITPACKV 2C User's Guide," Report CNA-191, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas.

[8] Kincaid, David R., Thomas C. Oppe, and David M. Young [1984] "Vector Computations for Sparse Linear Systems," Report CNA-189, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas.

[9] Kincaid, David R., and Thomas C. Oppe [1983] "ITPACK on Supercomputers," *Numerical Methods* (V. Pereyra and A. Reinoza, eds.) *Lecture Notes in Mathematics 1005*, Springer-Verlag, New York, pp. 151-161. (Also, Report CNA -178, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas, 1982.)

[10] Kincaid, David R., and David M. Young [1983] "The ITPACK Project: Past, Present, and Future," Report CNA-180, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas. (Also in *Elliptic Problem Solvers II* (Garrett Birkhoff and Arthur Schoenstadt, eds.), Academic Press, New York, pp. 53-63, 1984.)

[11] Kincaid, David R., John R. Respess, David M. Young, and Roger G. Grimes [1982] "ITPACK 2C: A Fortran Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods," *ACM Transactions on Mathematical Software*, Vol. 8, pp. 302-322.

[12] Kincaid, David R., Roger G. Grimes, and David M. Young [1979] "ITPACK — Adaptive Iterative Algorithms Using Symmetric Sparse Storage," in *Symposium on Reservoir Simulation*, Soc. Pet. Engr. of AIME, 6200 North Central Expressway, Dallas, Texas, pp. 151-160.

[13] Kincaid, David R., and David M. Young [1979] "Survey of Iterative Methods," in *Encyclopedia of Computer Sciences and Technology 13* (J. Belzer, A. Holzman, and A. Kent, eds.), Marcel Dekker, Inc., New York, pp. 354-391.

[14] Kincaid, David R., and Roger G. Grimes [1977] "ITPACK Report: Numerical Studies of Several Adaptive Iterative Algorithms," Report CNA-126, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas.

[15] Kincaid, David R., and David M. Young [1975] "The Development of a Computer Package for Solving a Class of Partial Differential Equations by Iterative Methods," Report CNA-102, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas. (Also in *Annales de l'Association Internationale pour le Calcul Analogique*, Vol. 3, pp. 186-191, 1975.)

[16] Mai, Tsun-Zee, [1986] "Adaptive Iterative Algorithms for Large Sparse Linear Systems," Report CNA-203, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas.

[17] Mai, Tsun-Zee, and David M. Young [1986] "ITPACK 3B User's Guide (Preliminary Version)," Report CNA-201, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas.

[18] Oppe, Thomas C., Wayne D. Joubert and David R. Kincaid [1988] "NSPCG User's Guide, Version 1.0, A Package for Solving Large Sparse Linear Systems by Various Iterative Methods," Report CNA-216, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas.

[19] Oppe, Thomas C., and David R. Kincaid [1987] "The Performance of ITPACK on Vector Computers for Solving Large Sparse Linear Sys-

tems Arising in Sample Oil Reservoir Problems," *Communications in Applied Numerical Methods*, Vol. 3, No. 1, pp 23-29.

[20] Rice, John R., and Ronald F. Boisvert [1985] *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York.

[21] Young, David M., [1987] "A Historical Review of Iterative Methods," Report CNA-206, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas. (To appear in *Proceedings of the Conference on the History of Scientific and Numeric Computing*, sponsored by the Association for Computing Machinery, held at Princeton, May 13-15, 1987.)

[22] Young, David M., Kang C. Jea, Tsun-Zee Mai [1987] "Preconditioned Conjugate Gradient Algorithms and Software for Solving Large Sparse Linear Systems, Report CNA-207, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas. (To appear in the Proceedings of the SIAM Conference on "Linear Algebra, Signals, and Control" which was held in Boston, Mass., August, 1986.)

[23] Young, David M., and Tsun-Zee Mai [1987] "Iterative Algorithms and Software for Solving Large Sparse Linear Systems," Report CNA-215, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas. (To appear in *Communications in Applied Numerical Methods*)

[24] Young, David M., and Tsun-Zee Mai [1984] "ITPACK 3A User's Guide (Preliminary Version)," Report CNA-197, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas.

[25] Young, David M., and David R. Kincaid [1980] "The ITPACK Package for Large Sparse Linear Systems," Report CNA-160, Center for Numerical Analysis, University of Texas at Austin, Austin, Texas. (Also in *Elliptic Problem Solvers* (M. Schultz, ed.), Academic Press, New York, pp. 163-185, 1981).

[26] Young, David M., [1971] "Iterative Solution of Large Linear Systems," Academic Press, New York.