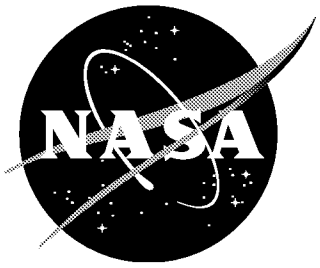
NASA/CR-2000-210537



User's Manual for PCSMS (Parallel Complex Sparse Matrix Solver)

Version 1.0

C. J. Reddy
Hampton University, Hampton, Virginia

October 2000

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

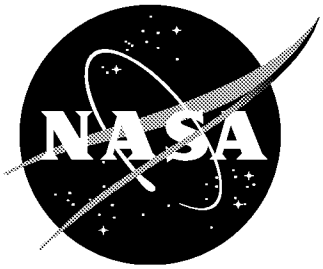
- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/CR-2000-210537



User's Manual for PCSMS (Parallel Complex Sparse Matrix Solver)

Version 1.0

C. J. Reddy
Hampton University, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Cooperative Agreement NCC1-366

October 2000

Available from:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 605-6000

CONTENTS

1. INTRODUCTION **1**

2. INSTALLATION OF THE CODE **3**

3. OPERATION OF THE CODE **3**

4. DRIVERS **8**

5. CONCLUDING REMARKS **8**

APPENDIX A: **10**

MATRIX STORAGE FORMATS 10

1. INTRODUCTION

Computational methods such as Finite Element Method (FEM) lead to sparse linear matrix equations, which are to be solved using either iterative solvers (conjugate gradient or biconjugate gradient, GMRES, QMR etc.) or using direct solvers (Gaussian Elimination, LU decomposition). Iterative solvers, though powerful, do not converge if the matrix is ill-conditioned. Also, if the matrix equation has to be solved for multiple right hand sides (for example, monostatic radar cross section (RCS) calculations for multiple incident angles), iterative solvers can be expensive in terms of CPU time as the matrix equation has to be solved for each incident angle. On the other hand, direct solvers tend to give a solution for all non-singular matrix equations. The sparse matrix is factored (or LU decomposed) once and can be used for solution, with multiple right-hand sides. The disadvantage with the direct solvers is that the sparse matrix gets filled during the factorization (LU decomposition) process and requires much more memory than originally needed. This disadvantage is alleviated by using reordering schemes, which rearrange the sparse matrix elements in such way that the factored matrix is filled only a few times over the original sparse matrix.

Disciplines such as computational structures, computational thermodynamics, computational fluid dynamics result in sparse matrices with real numbers. There are many direct sparse matrix solvers available both in public and commercial domains. Disciplines such as computational electromagnetics (CEM) and computational acoustics result in sparse matrices with complex numbers (symmetric and also unsymmetric). There are only a few direct sparse solvers available in the public domain (NETLIB at *www.netlib.org*, GAMS at *gams.nist.gov*) and only one commercially (CVSS at *www.solversoft.com*) to the best of this author's knowledge. Public domain direct complex sparse solvers suffer from the fact that

their reordering schemes are inefficient and result in large fill-ins in the sparse matrix and they are difficult to use. This led to our effort to use the existing real sparse direct solvers to solve complex, sparse matrix linear equations. A comprehensive library routine PCSMS (Parallel Complex Sparse Matrix Solver) was developed to convert complex matrices into real matrices and use SGI's `complib` routines to factor and solve the real matrices. The solution vector is reconverted to complex numbers. Though, this utility is written for SGI routines, it is general in nature and can be easily modified to work with any real sparse matrix solver. PCSMS (Parallel Complex Sparse Matrix Solver) is written based on SGI's sparse matrix routines for real matrices on SGI Origin 2000 to solve sparse linear systems of the form

$$[\mathrm{A}]\{x\} = \{b\} \tag{1}$$

PCSMS converts the complex sparse matrix equation to an equivalent real matrix equation and solves using the real sparse linear equation solvers based on SGI's complib library routines, PSLDLT (for symmetric sparse matrices) or PSLDU (for unsymmetric sparse matrices).

The User's Manual is written to make the user acquainted with the operation of PCSMS. The user is assumed to be familiar with SGI's operating environment. The organization of the manual is as follows. Section 2 explains the installation requirements. The operation of the code is given in detail in Section 3. Driver routines are presented in Section 4 for various matrix storage formats to aid the users to integrate PCSMS routines in their own codes. Sparse matrix storage formats are explained in Appendix A.

2. INSTALLATION OF THE CODE

The distribution disk of PCSMS contains a file named `pcsms.tar.gz`. The following commands can be used to get all the files.

```
gunzip pcsms.tar.gz

tar -xvf pcsms.tar
```

This creates a directory `PCSMS`, which in turn contains the subdirectories, `lib` (library module for `pcsms`), `Example1`, `Example2`, `Example3` and `Example4`. The code was tested on SGI Origin 2000 with the library `complib` containing the linear sparse matrix routines, `PSLDLT` and `PSLDU`.

3. OPERATION OF THE CODE

The `PCSMS` directories created in the above section can be located in a directory accessible to all users, if the `PCSMS` is to be accessible to all users on the system. The library can be linked to the main code by linking `PCSMSLIB/lib/pcsms.a`, where `PCSMSLIB` is the path, where `PCSMS` directory is located.

The following routines are included in `pcsms.a`

`CSSMS1` - Complex Symmetric Sparse Matrix Solver (Simple Storage format)

`CSSMS2` - Complex Symmetric Sparse Matrix Solver (CCS format)

`CGSMS1` - Complex Unsymmetric Sparse Matrix Solver (Simple format)

`CGSMS2` - Complex Unsymmetric Sparse Matrix Solver (CCS format)

Syntax for calling the above routines is given below:

`CSSMS1` - Complex Symmetric Sparse Matrix Solver (Simple Storage format):

```
CALL CSSMS1 (A, IR, JC, N, NZ, MAXN, MAXNZ, B, X, NORDER,
            INDC)
COMPLEX A(MAXNZ), B(MAXN), X(MAXN)
INTEGER IR(MAXNZ), JC(MAXNZ)
```

```
INTEGER N, NZ, NORDER, INDC
```

Input:

A - Complex symmetric matrix **A** stored as a single indexed array, only the lower half of the matrix is stored.

B - Right hand side vector **b**

X - Solution vector **x**

IR - Array storing the row indices of the values in A. Row index of A(i) is given by IR(i).

JC - Array storing the column indices of the values of A. Column index of A(i) is given by JC(i).

N - Number of Unknowns.

NZ - Number of non-zero values in the lower half of the matrix **A**.

MAXN - Maximum number of Unknowns (>NZ)

MAXNZ - Maximum number of non-zero values (>NZ)

NORDER - Reordering algorithm to be used.

NORDER=0	No reordering is done (NOT RECOMMENDED)
NORDER=1	Approximate Minimum Degree Reordering is done
NORDER=2	Multi-level Nested Dissection Reordering is done

INDC - INDC takes the values 1, 2 or 3.

INDC=1	LU Factor the matrix **A**
INDC=2	Backsolve the matrix equation **Ax=b**
INDC=3	Free all the temporary memory used.

Once the matrix is LU factored, with INDC=1, the matrix equation can be backsolved many times by calling CSSMS1 with INDC=2. After all the backsolves are completed, the temporary memory can be released by calling CSSMS1 with INDC=3.

CSSMS2 - Complex Symmetric Sparse Matrix Solver (CCS format):

```
CALL CSSMS2(A, IR, JC, N, NZ, MAXN, MAXNZ, B, X, NORDER, INDC)
COMPLEX A(MAXNZ), B(MAXN), X(MAXN)
INTEGER IR(MAXNZ), JC(MAXN)
INTEGER N, NZ, NORDER, INDC
```

Input:

A - Complex symmetric matrix **A** stored as a single indexed array, only the lower half of the matrix is stored.

B - Right hand side vector **b**

X - Solution vector **x**

IR – Array storing the row indices of the values in A. Row index of A(i) is given by IR(i).

JC – Array storing the index in IR() for the first non-zero in each column of lower half of matrix **A**. The row indices for the non-zeros in column i can be found in locations IR(JC(i)) through IR(JC(i+1)-1). The corresponding non-zero values can be found in locations A(JC(i)) through A(JC(i+1)-1). The array contains (N+1) entries.

N - Number of Unknowns.

NZ - Number of non-zero values in the lower half of the matrix **A**.

MAXN – Maximum number of Unknowns (>NZ+1)

MAXNZ - Maximum number of non-zero values (>NZ)

NORDER - Reordering algorithm to be used.

NORDER=0 No reordering is done (NOT RECOMMENDED)
NORDER=1 Approximate Minimum Degree Reordering is done
NORDER=2 Multi-level Nested Dissection Reordering is done

INDC - INDC takes the values 1, 2 or 3.

INDC=1	LU Factor the matrix A
INDC=2	Backsolve the matrix equation Ax=b
INDC=3	Free all the temporary memory used.

Once the matrix is LU factored, with INDC=1, the matrix equation can be backsolved many times by calling CSSMS2 with INDC=2. After all the backsolves are completed, the temporary memory can be released by calling CSSMS2 with INDC=3.

CGSMS1 - Complex Unsymmetric Sparse Matrix Solver (Simple format):

```
CALL CGSMS1(A, IR, JC, N, NZ, MAXN, MAXNZ, B, X, NORDER, INDC)
COMPLEX A(MAXNZ), B(MAXN), X(MAXN)
INTEGER IR(MAXNZ), JC(MAXNZ)
INTEGER N, NZ, NORDER, INDC
```

Input:

A - Complex unsymmetric matrix **A** stored as a single indexed array.

B - Right hand side vector **b**

X - Solution vector **x**

IR - Array storing the row indices of the values in A. Row index of A(i) is given by IR(i).

JC - Array storing the column indices of the values of A. Column index of A(i) is given by JC(i).

N - Number of Unknowns.

NZ - Number of non-zero values in matrix **A**.

MAXN - Maximum number of Unknowns (>NZ)

MAXNZ - Maximum number of non-zero values (>NZ)

NORDER - Reordering algorithm to be used.

NORDER=0 No reordering is done (NOT RECOMMENDED)
NORDER=1 Approximate Minimum Degree Reordering is done
NORDER=2 Multi-level Nested Dissection Reordering is done

INDC - INDC takes the values 1, 2 or 3.

INDC=1	LU Factor the matrix **A**
INDC=2	Backsolve the matrix equation **Ax=b**
INDC=3	Free all the temporary memory used.

Once the matrix is LU factored, with INDC=1, the matrix equation can be backsolved many times by calling CGSMS1 with INDC=2. After all the backsolves are completed, the temporary memory can be released by calling CGSMS1 with INDC=3.

CGSMS2 - Complex Unsymmetric Sparse Matrix Solver (CCS format):

```
CALL CGSMS2(A, IR, JC, N, NZ, MAXN, MAXNZ, B, X, NORDER, INDC)
COMPLEX A(MAXNZ), B(MAXN), X(MAXN)
INTEGER IR(MAXNZ), JC(MAXN)
INTEGER N, NZ, NORDER, INDC
```

Input:

A - Complex unsymmetric matrix **A** stored as a single indexed array.

B - Right hand side vector **b**

X - Solution vector **x**

IR - Array storing the row indices of the values in A. Row index of A(i) is given by IR(i).

JC - Array storing the index in IR() for the first non-zero in each column of lower half of matrix **A**. The row indices for the non-zeros in column i can be found in locations IR(JC(i)) through IR(JC(i+1)-1). The corresponding non-zero values can be found in locations A(JC(i)) through A(JC(i+1)-1). The array contains (N+1) entries.

N - Number of Unknowns.

NZ - Number of non-zero values in matrix **A**.

MAXN - Maximum number of Unknowns (>NZ+1)

MAXNZ - Maximum number of non-zero values (>NZ)

NORDER - Reordering algorithm to be used.

NORDER=0 No reordering is done (NOT RECOMMENDED)
NORDER=1 Approximate Minimum Degree Reordering is done
NORDER=2 Multi-level Nested Dissection Reordering is done

INDC - INDC takes the values 1, 2 or 3.

INDC=1	LU Factor the matrix A
INDC=2	Backsolve the matrix equation Ax=b
INDC=3	Free all the temporary memory used.

Once the matrix is LU factored, with INDC=1, the matrix equation can be backsolved many times by calling CGSMS2 with INDC=2. After all the backsolves are completed, the temporary memory can be released by calling CGSMS2 with INDC=3.

4. Drivers

Drivers are provided as an illustration of interfacing the matrices with the PCSMS routines. The directory Example1 contains the driver for cssms1 (driver_cssms1.f), Example2 directory contains the driver for cssms2 (driver_cssms2). Driver for cgsms1 (driver_cgsms1) is in Example3 directory and the driver for cgsms2 (driver_cgsms2) is in Example4 directory. Users can use the driver routines to integrate PCSMS library into their own codes.

5. Concluding Remarks

PCSMS library is built to solve the complex sparse matrix equations using SGI's real sparse matrix solvers. This library is used with both symmetric and unsymmetric complex sparse matrices. Two different matrix storage formats, simple storage and compressed column storage formats are available to be used with PCSMS library. With proper translators, any other storage formats can be incorporated. In the future, PCSMS library can be modified to

include public domain real sparse matrix solvers to make it portable to various computer systems, such as SUN's Solaris OS, PC's Windows OS and Linux OS.

As with any software, PCSMS may have some bugs and need improvement. Please contact

Electromagnetic Research Branch
Mail Stop 490
NASA Langley Research Center
Hampton VA 23681
Phone: 757.864.1772
Fax: 757.864.7975

with any bug reports, suggestions and comments.

Appendix A:

Matrix Storage Formats

Sparse matrices contain some non-zero values and many zero values. It is advantageous to store the matrix in a format that will store only the non-zero values of the matrix and hence saving computer storage. In this report, two different sparse matrix formats (1) Simple Storage format and (2) Compressed Column Storage format, are used. User can choose either of the formats and call the appropriate library routine to solve the matrix.

As an example, the following sparse matrix is used to illustrate the above two sparse matrix storage formats[1].

$$[A] = \begin{bmatrix} 11 & 0 & 13 & 14 & 0 & 0 \\ 0 & 22 & 23 & 0 & 25 & 0 \\ 31 & 32 & 33 & 0 & 35 & 0 \\ 41 & 0 & 0 & 44 & 45 & 0 \\ 0 & 52 & 53 & 54 & 55 & 0 \\ 0 & 0 & 0 & 0 & 0 & 66 \end{bmatrix}$$

Simple Storage Format:

When the matrix [A] is stored in simple storage format, the following arrays are used.

`IR` – Integer array used to store the row indices of the non-zero values in matrix [A].

`JC` – Integer array used to store the corresponding column indices of the non-zero values in matrix [A].

`A` – Array storing the corresponding non-zero value, indicated by `IR` and `JC` array entries.

For the above matrix:

`IR` = {1 3 2 4 1 3 2 2 1 4 5 5 3 5 6 4 5 3}

`JC` = {1 1 2 1 3 2 3 5 4 4 2 3 5 5 6 5 4 3}

`A` = {11 31 22 41 13 32 23 25 14 44 52 53 35 55 66 45 54 33}

Note that the row and column indices of `A(i)` are stored in `IR(i)` and `JC(i)` respectively. The entries in `A` need not be in any particular order. For symmetric matrices only the non-zero entries in lower half of the matrix [A] are stored.

[1] For illustration purpose the matrix is chosen to be real. The storage format remains the same for complex matrix also.

Compressed Column Storage Format:

In simple storage format the JC array has repeated entries. In compressed column storage format the JC array is replaced by a shorter array that gives the index of the first element of each column in IR and A. The row indices for the non-zeros in column i can be found in locations IR(JC(i)) through IR(JC(i+1)-1). The corresponding non-zero values can be found in locations A(JC(i)) through A(JC(i+1)-1). The array JC contains (N+1) entries.

JC = {1 4 7 11 14 18 19}

IR = {1 3 4 2 3 5 1 2 3 5 1 4 5 2 3 4 5 6}

A = {11 31 41 22 32 52 13 23 33 53 14 44 54 25 35 45 55 66}

For example, the non-zero entries in column 3 can be extracted as below:

Column 3 has 4 non-zero entries JC(4)-JC(3) = 11-7

The row indices of non-zero entries in column 3 are given by IR(7) to IR(10) - 1 2 3 5

Similarly the corresponding non-zero values in column 3 are given by A(7) to A(10) - 13 23 33 55.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 2000	Contractor Report

4. TITLE AND SUBTITLE
User's Manual for PCSMS (Parallel Complex Sparse Matrix Solver)
Version 1.0

5. FUNDING NUMBERS
WU 522-31-21
NCC1-366

6. AUTHOR(S)
C. J. Reddy

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Hampton University
Hampton, VA 23668

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-2199

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/CR-2000-210537

11. SUPPLEMENTARY NOTES
NASA Contract Monitor: M. C. Bailey

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 04 Distribution: Nonstandard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

PCSMS (Parallel Complex Sparse Matrix Solver) is a computer code written to make use of the existing real sparse direct solvers to solve complex, sparse matrix linear equations. PCSMS converts complex matrices into real matrices and use real, sparse direct matrix solvers to factor and solve the real matrices. The solution vector is reconverted to complex numbers. Though, this utility is written for Silicon Graphics (SGI) real sparse matrix solution routines, it is general in nature and can be easily modified to work with any real sparse matrix solver. The User's Manual is written to make the user acquainted with the installation and operation of the code. Driver routines are given to aid the users to integrate PCSMS routines in their own codes.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Sparse Matrix Solvers, Direct Methods, Matrix Solvers, Complex Matrices	17
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL