Symmetry Reductions

Dr. George W Benthien

January 16, 2001

E-mail: george@gbenthien.net

Contents

1 Introduction

The numerical approximation of many physical problems leads to the solution of a system of linear algebraic equations of the form

$$Ax = b \tag{1}$$

where A is an $N \times N$ matrix representing some linear operator, b is an N-vector representing a forcing function, and x is an N-vector of unknowns corresponding to the values of some scalar physical quantity at N locations in a spatial region of interest. For example, the components of x might correspond to the values of pressure or electric potential at a discrete set of points. In this paper we have restricted our attention to problems involving scalar physical variables, but the general techniques developed can be extended to problems involving vector variables by appropriately extending the definition of the symmetry operators. Symmetry in the physical problem manifests itself in the structure of the matrix A. For example, with proper numbering of the evaluation points, one plane of symmetry leads to the following structure of A

$$A = \begin{pmatrix} A_1 & A_2 \\ A_2 & A_1 \end{pmatrix}. \tag{2}$$

Here A_1 and A_2 are submatrices of A that are one-half the size of A. The special forms resulting from various types of symmetry can be utilized to significantly reduce the solution time of equation (1). In this paper we will consider the following types of symmetry

1. one, two, or three planes of symmetry
2. any finite order of rotational symmetry
3. any finite order of rotational symmetry plus one additional plane of symmetry.

It is not necessary for the forcing vector b to have the same symmetry as the rest of the problem in order to achieve reductions in solution times, but additional reductions can be achieved if the forcing vector has the same symmetry. Table 1 below shows the reductions in computational times that can be obtained by taking advantage of these symmetries.

We have implemented these symmetry reductions as a series of Fortran subroutines that are contained in a Fortran 90 module. The subroutines in this module are described at the end of this paper.

Table 1: Time reduction factors for various types of symmetry

Symmetry Type	Solution	Solution with right-hand-side symmetry	Matrix generation
1 plane of symmetry	4	8	2
2 planes of symmetry	16	64	4
3 planes of symmetry	64	512	8
N-fold rotational symmetry	N^2	N^3	N
N-fold rotational symmetry plus 1 plane of symmetry	$4N^2$	$8N^3$	$2N$

2 Basic Theory

In this section we will discuss the basic theory behind the symmetry reduction schemes that are the subject of this paper. All of the symmetry reductions we will consider are based on two simple concepts. The first is that the symmetry in the physical problem manifests itself in the commutation of the operator A in equation (3) with certain symmetry operators (reflections and/or rotations). The second is a result from linear algebra stating that the eigenspaces of an operator are invariant under any operator that commutes with it. In the remainder of this section we will apply these concepts to the various types of symmetry under consideration.

2.1 One plane of symmetry

Consider a planar region as shown in figure 1 that has one plane of symmetry. The evaluation points are numbered from 1 to 16.

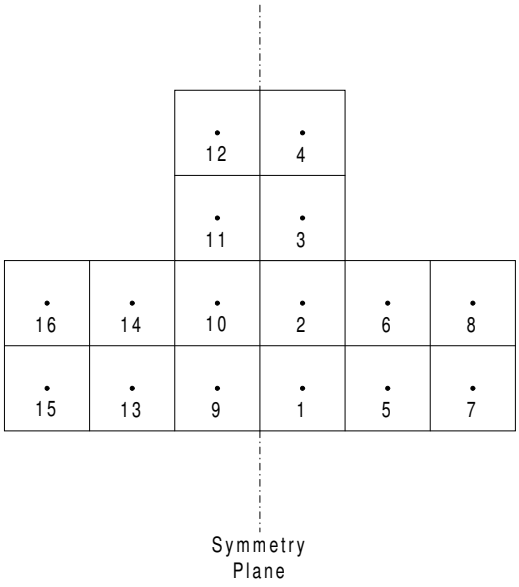

Figure 1: Subdivided region with one plane of symmetry

Notice that the evaluation points are symmetrically located relative to the symmetry plane and that the numbering of points in symmetric portions of the region is in the same order. This symmetric numbering greatly simplifies the resulting matrix structure. Suppose that the physics of the problem leads to a system of linear equations

$$Ax = b \tag{3}$$

for the unknown vector x, where the components of x correspond to the values of some scalar physical variable at the evaluation points shown. Let x be partitioned into two parts corresponding to the two sides of the symmetry plane as follows

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \tag{4}$$

The operator Σ corresponding to reflection across the symmetry plane is defined by

$$\Sigma x = \Sigma \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} \qquad \text{for all } x. \tag{5}$$

It is easily verified that Σ must have the block form

$$\Sigma = \begin{pmatrix} 0 & \mathrm{I} \\ \mathrm{I} & 0 \end{pmatrix} \tag{6}$$

where I is an identity matrix. Suppose that the forcing vector b is reflected across the symmetry plane, i.e., b is replaced by Σb. The symmetry of the problem implies that the resulting solution x would also be reflected across the symmetry plane, i.e., x becomes Σx. For example, the response at location 11 due to a unit forcing function at location 4 is the same as the response at location 3 due to a unit forcing function at location 12. Thus, in general

$$A\Sigma x = \Sigma b = \Sigma A x. \tag{7}$$

Since equation (7) must hold for any choice of b (and hence x), it follows that

$$A\Sigma = \Sigma A, \tag{8}$$

i.e., the operators Σ and A commute. Substituting the partitioned forms of A and Σ into equation (8) and carrying out the block multiplications shows that A has the block form

$$A = \begin{pmatrix} A_1 & A_2 \\ A_2 & A_1 \end{pmatrix}. \tag{9}$$

Commuting operators have the property that the eigenspaces of one of the operators are invariant under the other operator. For example, if e is an eigenvector of Σ corresponding to the eigenvalue λ, then

$$\Sigma(Ae) = A(\Sigma e) = \lambda(Ae), \tag{10}$$

i.e., Ae is also an eigenvector of Σ corresponding to the eigenvalue λ. Since symmetry operators like Σ have relatively simple eigenstructures that can be determined by inspection or simple analysis, we will use the fact that these eigenspaces are invariant under A. The eigenvalues of Σ are ± 1. A basis of eigenvectors corresponding to the eigenvalue +1 is given by the columns of the matrix $\begin{pmatrix} \mathrm{I} \\ \mathrm{I} \end{pmatrix}$ where I is the identity matrix. Similarly, a basis of eigenvectors corresponding to -1 is given by $\begin{pmatrix} \mathrm{I} \\ -\mathrm{I} \end{pmatrix}$. If we change to the basis given by the columns of

$$S = \begin{pmatrix} \mathrm{I} & \mathrm{I} \\ \mathrm{I} & -\mathrm{I} \end{pmatrix} \tag{11}$$

then the invariance of the eigenspaces of Σ under A implies that the matrix will become block diagonal relative to this basis. Under this change of basis A becomes $S^{-1}AS$. The matrix S has the property

$$S^{-1} = \tfrac{1}{2}S. \tag{12}$$

Thus,

$$S^{-1}AS = \tfrac{1}{2}SAS = \begin{pmatrix} A_1 + A_2 & 0 \\ 0 & A_1 - A_2 \end{pmatrix} \tag{13}$$

and the system of equations (3) becomes

$$(S^{-1}AS)(S^{-1}x) = \begin{pmatrix} A_1 + A_2 & 0 \\ 0 & A_1 - A_2 \end{pmatrix} \begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \end{pmatrix} = S^{-1}b = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix} \tag{14}$$

where

$$\hat{x} = \begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \end{pmatrix} = S^{-1}x. \tag{15}$$

Equations (14)–(15) imply that the original system of equations (3) can be replaced by two systems of half the size, i.e.,

$$\begin{aligned} (A_1 + A_2)\,\hat{x}_1 &= \hat{b}_1 \\ (A_1 - A_2)\,\hat{x}_2 &= \hat{b}_2 \end{aligned} \tag{16a}$$

where

$$\begin{aligned} \hat{b}_1 &= \tfrac{1}{2}(b_1 + b_2) \\ \hat{b}_2 &= \tfrac{1}{2}(b_1 - b_2) \end{aligned} \tag{16b}$$

and

$$\begin{aligned} x_1 &= \hat{x}_1 + \hat{x}_2 \\ x_2 &= \hat{x}_1 - \hat{x}_2. \end{aligned} \tag{16c}$$

Since solution time is proportional to the cube of the number of equations, it is four times faster to solve the two smaller systems than the original larger system. If the right-hand-side b is symmetric across the symmetry plane, then it follows that $b_1 = b_2$, $\hat{b}_1 = b_1$, $\hat{b}_2 = 0$, $\hat{x}_2 = 0$, and $x_1 = x_2 = \hat{x}_1$. In this case it is only necessary to solve the single system of equations

$$(A_1 + A_2)\, x_1 = b_1 \tag{17}$$

and set $x_2 = x_1$.

2.2 Two planes of symmetry

Consider next a problem having two planes of symmetry. The four symmetry quadrants will be numbered as shown in table 2.

Table 2: Numbering of quadrants for two planes of symmetry

Quadrant	x_1	x_2
1	+	+
2	+	-
3	-	+
4	-	-

Two planes of symmetry can be handled by applying the results for one plane of symmetry successively to the two reflection operators. If the evaluation points are numbered symmetrically in the four symmetry quadrants, then the two reflection operators Σ_1 and Σ_2 can be written

$$\Sigma_1 = \begin{pmatrix} 0 & 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 & \mathrm{I} \\ \mathrm{I} & 0 & 0 & 0 \\ 0 & \mathrm{I} & 0 & 0 \end{pmatrix} \qquad \Sigma_2 = \begin{pmatrix} 0 & \mathrm{I} & 0 & 0 \\ \mathrm{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{I} \\ 0 & 0 & \mathrm{I} & 0 \end{pmatrix}. \tag{18}$$

Notice that

$$\Sigma_1 \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ x_1 \\ x_2 \end{pmatrix} \qquad \text{and} \qquad \Sigma_2 \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_1 \\ x_4 \\ x_3 \end{pmatrix}. \tag{19}$$

The fact that A commutes with both Σ_1 and Σ_2 implies that A must have the block structure

$$A = \begin{pmatrix} A_1 & A_2 & A_3 & A_4 \\ A_2 & A_1 & A_4 & A_3 \\ A_3 & A_4 & A_1 & A_2 \\ A_4 & A_3 & A_2 & A_1 \end{pmatrix}. \tag{20}$$

The eigenvector matrices S_1 and S_2 corresponding to Σ_1 and Σ_2 are

$$S_1 = \begin{pmatrix} \mathrm{I} & 0 & \mathrm{I} & 0 \\ 0 & \mathrm{I} & 0 & \mathrm{I} \\ \mathrm{I} & 0 & -\mathrm{I} & 0 \\ 0 & \mathrm{I} & 0 & -\mathrm{I} \end{pmatrix} \qquad S_2 = \begin{pmatrix} \mathrm{I} & \mathrm{I} & 0 & 0 \\ \mathrm{I} & -\mathrm{I} & 0 & 0 \\ 0 & 0 & \mathrm{I} & \mathrm{I} \\ 0 & 0 & \mathrm{I} & -\mathrm{I} \end{pmatrix}. \tag{21}$$

If we apply the change of basis corresponding to S_1 followed by the change of basis corresponding to S_2, then A becomes $S_2^{-1}S_1^{-1}AS_1S_2$ or $(S_1S_2)^{-1}A(S_1S_2)$. Thus, the matrix S_1S_2 plays the same role for two planes of symmetry as S did for one plane of symmetry. As before, this change of basis makes A block diagonal. It is easily verified that

$$S_1S_2 = \begin{pmatrix} \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} \\ \mathrm{I} & -\mathrm{I} & \mathrm{I} & -\mathrm{I} \\ \mathrm{I} & \mathrm{I} & -\mathrm{I} & -\mathrm{I} \\ \mathrm{I} & -\mathrm{I} & -\mathrm{I} & \mathrm{I} \end{pmatrix} \tag{22}$$

and that the diagonal submatrices of $(S_1S_2)^{-1}A(S_1S_2)$ are $A_1 + A_2 + A_3 + A_4$, $A_1 - A_2 + A_3 - A_4$, $A_1 + A_2 - A_3 - A_4$, and $A_1 - A_2 - A_3 + A_4$. Thus, two planes of symmetry allow the original system of equations to be replaced by four smaller systems of one-quarter the size, i.e.,

$$\begin{aligned} (A_1 + A_2 + A_3 + A_4)\,\hat{x}_1 &= \hat{b}_1 \\ (A_1 - A_2 + A_3 - A_4)\,\hat{x}_2 &= \hat{b}_2 \\ (A_1 + A_2 - A_3 - A_4)\,\hat{x}_3 &= \hat{b}_3 \\ (A_1 - A_2 - A_3 + A_4)\,\hat{x}_4 &= \hat{b}_4 \end{aligned} \tag{23a}$$

where

$$\begin{aligned} \hat{b}_1 &= \tfrac{1}{4}(b_1 + b_2 + b_3 + b_4) \\ \hat{b}_2 &= \tfrac{1}{4}(b_1 - b_2 + b_3 - b_4) \\ \hat{b}_3 &= \tfrac{1}{4}(b_1 + b_2 - b_3 - b_4) \\ \hat{b}_4 &= \tfrac{1}{4}(b_1 - b_2 - b_3 + b_4) \end{aligned} \tag{23b}$$

and

$$\begin{aligned} x_1 &= \hat{x}_1 + \hat{x}_2 + \hat{x}_3 + \hat{x}_4 \\ x_2 &= \hat{x}_1 - \hat{x}_2 + \hat{x}_3 - \hat{x}_4 \\ x_3 &= \hat{x}_1 + \hat{x}_2 - \hat{x}_3 - \hat{x}_4 \\ x_4 &= \hat{x}_1 - \hat{x}_2 - \hat{x}_3 + \hat{x}_4. \end{aligned} \tag{23c}$$

If the right-hand-side also has two planes of symmetry, then it is only necessary to solve the single system

$$(A_1 + A_2 + A_3 + A_4)\, x_1 = b_1 \tag{24}$$

and set $x_2 = x_3 = x_4 = x_1$.

2.3 Three planes of symmetry

Consider next a problem having three planes of symmetry. We will number the symmetry octants as shown in table 3.

Table 3: Numbering of octants for three planes of symmetry

Octant	x_1	x_2	x_3
1	+	+	+
2	+	+	-
3	+	-	+
4	+	-	-
5	-	+	+
6	-	+	-
7	-	-	+
8	-	-	-

In this case we have three reflection operators

$$\Sigma_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} \\ \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 \end{pmatrix} \tag{25a}$$

$$\Sigma_2 = \begin{pmatrix} 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 \\ \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} \\ 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 \end{pmatrix} \tag{25b}$$

$$\Sigma_3 = \begin{pmatrix} 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 \end{pmatrix}. \tag{25c}$$

The three eigenvector matrices are

$$S_1 = \begin{pmatrix} \mathrm{I} & 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 \\ 0 & \mathrm{I} & 0 & 0 & 0 & \mathrm{I} & 0 & 0 \\ 0 & 0 & \mathrm{I} & 0 & 0 & 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & \mathrm{I} \\ \mathrm{I} & 0 & 0 & 0 & -\mathrm{I} & 0 & 0 & 0 \\ 0 & \mathrm{I} & 0 & 0 & 0 & -\mathrm{I} & 0 & 0 \\ 0 & 0 & \mathrm{I} & 0 & 0 & 0 & -\mathrm{I} & 0 \\ 0 & 0 & 0 & \mathrm{I} & 0 & 0 & 0 & -\mathrm{I} \end{pmatrix} \tag{26}$$

$$S_2 = \begin{pmatrix} \mathrm{I} & 0 & \mathrm{I} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{I} & 0 & \mathrm{I} & 0 & 0 & 0 & 0 \\ \mathrm{I} & 0 & -\mathrm{I} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{I} & 0 & -\mathrm{I} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{I} & 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & \mathrm{I} \\ 0 & 0 & 0 & 0 & \mathrm{I} & 0 & -\mathrm{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{I} & 0 & -\mathrm{I} \end{pmatrix} \tag{27}$$

$$S_3 = \begin{pmatrix} \mathrm{I} & \mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathrm{I} & -\mathrm{I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathrm{I} & \mathrm{I} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathrm{I} & -\mathrm{I} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{I} & \mathrm{I} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{I} & -\mathrm{I} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & \mathrm{I} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{I} & -\mathrm{I} \end{pmatrix}. \tag{28}$$

The product $S_1 S_2 S_3$ is the matrix of basis vectors that block diagonalizes A. It is given by

$$S_1 S_2 S_3 = \begin{pmatrix} \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} \\ \mathrm{I} & -\mathrm{I} & \mathrm{I} & -\mathrm{I} & \mathrm{I} & -\mathrm{I} & \mathrm{I} & -\mathrm{I} \\ \mathrm{I} & \mathrm{I} & -\mathrm{I} & -\mathrm{I} & \mathrm{I} & \mathrm{I} & -\mathrm{I} & -\mathrm{I} \\ \mathrm{I} & -\mathrm{I} & -\mathrm{I} & \mathrm{I} & \mathrm{I} & -\mathrm{I} & -\mathrm{I} & \mathrm{I} \\ \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{I} & -\mathrm{I} & -\mathrm{I} & -\mathrm{I} & -\mathrm{I} \\ \mathrm{I} & -\mathrm{I} & \mathrm{I} & -\mathrm{I} & -\mathrm{I} & \mathrm{I} & -\mathrm{I} & \mathrm{I} \\ \mathrm{I} & \mathrm{I} & -\mathrm{I} & -\mathrm{I} & -\mathrm{I} & -\mathrm{I} & \mathrm{I} & \mathrm{I} \\ \mathrm{I} & -\mathrm{I} & -\mathrm{I} & \mathrm{I} & -\mathrm{I} & \mathrm{I} & \mathrm{I} & -\mathrm{I} \end{pmatrix}. \tag{29}$$

In this case we can reduce the original system of equations to eight systems of one-eighth the size, i.e.,

$$\begin{aligned} (A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8)\,\hat{x}_1 &= \hat{b}_1 \\ (A_1 - A_2 + A_3 - A_4 + A_5 - A_6 + A_7 - A_8)\,\hat{x}_2 &= \hat{b}_2 \\ (A_1 + A_2 - A_3 - A_4 + A_5 + A_6 - A_7 - A_8)\,\hat{x}_3 &= \hat{b}_3 \\ (A_1 - A_2 - A_3 + A_4 + A_5 - A_6 - A_7 + A_8)\,\hat{x}_4 &= \hat{b}_4 \\ (A_1 + A_2 + A_3 + A_4 - A_5 - A_6 - A_7 - A_8)\,\hat{x}_5 &= \hat{b}_5 \\ (A_1 - A_2 + A_3 - A_4 - A_5 + A_6 + A_7 - A_8)\,\hat{x}_6 &= \hat{b}_6 \\ (A_1 + A_2 - A_3 - A_4 - A_5 - A_6 + A_7 + A_8)\,\hat{x}_7 &= \hat{b}_7 \\ (A_1 - A_2 - A_3 + A_4 - A_5 + A_6 + A_7 - A_8)\,\hat{x}_8 &= \hat{b}_8 \end{aligned} \tag{30a}$$

where

$$\begin{aligned} \hat{b}_1 &= \tfrac{1}{8}(b_1 + b_2 + b_3 + b_4 + b_5 + b_6 + b_7 + b_8) \\ \hat{b}_2 &= \tfrac{1}{8}(b_1 - b_2 + b_3 - b_4 + b_5 - b_6 + b_7 - b_8) \\ \hat{b}_3 &= \tfrac{1}{8}(b_1 + b_2 - b_3 - b_4 + b_5 + b_6 - b_7 - b_8) \\ \hat{b}_4 &= \tfrac{1}{8}(b_1 - b_2 - b_3 + b_4 + b_5 - b_6 - b_7 + b_8) \\ \hat{b}_5 &= \tfrac{1}{8}(b_1 + b_2 + b_3 + b_4 - b_5 - b_6 - b_7 - b_8) \\ \hat{b}_6 &= \tfrac{1}{8}(b_1 - b_2 + b_3 - b_4 - b_5 + b_6 - b_7 + b_8) \\ \hat{b}_7 &= \tfrac{1}{8}(b_1 + b_2 - b_3 - b_4 - b_5 - b_6 + b_7 + b_8) \\ \hat{b}_8 &= \tfrac{1}{8}(b_1 - b_2 - b_3 + b_4 - b_5 + b_6 + b_7 - b_8) \end{aligned} \tag{30b}$$

and

$$
\begin{aligned}
x_1 &= \hat{x}_1 + \hat{x}_2 + \hat{x}_3 + \hat{x}_4 + \hat{x}_5 + \hat{x}_6 + \hat{x}_7 + \hat{x}_8 \\
x_2 &= \hat{x}_1 - \hat{x}_2 + \hat{x}_3 - \hat{x}_4 + \hat{x}_5 - \hat{x}_6 + \hat{x}_7 - \hat{x}_8 \\
x_3 &= \hat{x}_1 + \hat{x}_2 - \hat{x}_3 - \hat{x}_4 + \hat{x}_5 + \hat{x}_6 - \hat{x}_7 - \hat{x}_8 \\
x_4 &= \hat{x}_1 - \hat{x}_2 - \hat{x}_3 + \hat{x}_4 + \hat{x}_5 - \hat{x}_6 - \hat{x}_7 + \hat{x}_8 \\
x_5 &= \hat{x}_1 + \hat{x}_2 + \hat{x}_3 + \hat{x}_4 - \hat{x}_5 - \hat{x}_6 - \hat{x}_7 - \hat{x}_8 \\
x_6 &= \hat{x}_1 - \hat{x}_2 + \hat{x}_3 - \hat{x}_4 - \hat{x}_5 + \hat{x}_6 - \hat{x}_7 + \hat{x}_8 \\
x_7 &= \hat{x}_1 + \hat{x}_2 - \hat{x}_3 - \hat{x}_4 - \hat{x}_5 - \hat{x}_6 + \hat{x}_7 + \hat{x}_8 \\
x_8 &= \hat{x}_1 - \hat{x}_2 - \hat{x}_3 + \hat{x}_4 - \hat{x}_5 + \hat{x}_6 + \hat{x}_7 - \hat{x}_8.
\end{aligned}
\tag{30c}
$$

If the right-hand-side also has three planes of symmetry, then it is only necessary to solve the single system

$$
(A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8)\, x_1 = b_1 \tag{31}
$$

and set $x_2 = x_3 = x_4 = x_5 = x_6 = x_7 = x_8 = x_1$.

2.4 Finite order rotational symmetry

We will consider next finite order rotational symmetry. For example, a pentagon has rotational symmetry of order five and an octagon has rotational symmetry of order eight. As before we will assume that each symmetry block is numbered in the same order. The rotation operator R has the property

$$Rx = R\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} x_2 \\ x_3 \\ \vdots \\ x_N \\ x_1 \end{pmatrix} \qquad \text{for all } x. \tag{32}$$

Thus, R must have the block form

$$R = \begin{pmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & I \\ I & 0 & 0 & \cdots & 0 \end{pmatrix}. \tag{33}$$

The eigenvalues of R are the N-th roots of unity. The eigenvector matrix S is given by

$$S = \begin{pmatrix} S_{11} & \cdots & S_{1N} \\ \vdots & & \vdots \\ S_{N1} & \cdots & S_{NN} \end{pmatrix}. \tag{34}$$

where the block S_{mn} is defined by

$$S_{mn} = e^{i\frac{2\pi(m-1)(n-1)}{N}} I. \tag{35}$$

The inverse of S is given by

$$S^{-1} = \frac{1}{N}\begin{pmatrix} S_{11}^* & \cdots & S_{1N}^* \\ \vdots & & \vdots \\ S_{N1}^* & \cdots & S_{NN}^* \end{pmatrix} \tag{36}$$

where $*$ denotes the complex conjugate.

The rotational symmetry implies that A commutes with R. This commutation relation implies that A has the block circulant form

$$A = \begin{pmatrix} A_1 & A_2 & \cdots & A_N \\ A_N & A_1 & \cdots & A_{N-1} \\ \vdots & & & \vdots \\ A_2 & \cdots & A_N & A_1 \end{pmatrix}. \tag{37}$$

To see this multiply out the blocks of RA and AR as before. The commutation of A with R also implies that $S^{-1}AS$ is block diagonal. The diagonal blocks are given by

$$[S^{-1}AS]_{mm} = \sum_{n=1}^{N} e^{i\frac{2\pi(m-1)(n-1)}{N}} A_n \qquad m = 1, \ldots, N. \tag{38}$$

Thus, the original system of equations reduces to the N smaller systems of equations

$$\left(\sum_{n=1}^{N} e^{i\frac{2\pi(m-1)(n-1)}{N}} A_n \right) \hat{x}_m = \hat{b}_m \qquad m = 1, \ldots, N \tag{39a}$$

where

$$\hat{b}_m = \frac{1}{N} \sum_{n=1}^{N} e^{-i\frac{2\pi(m-1)(n-1)}{N}} b_n \tag{39b}$$

and

$$x_m = \sum_{n=1}^{N} e^{i\frac{2\pi(m-1)(n-1)}{N}} \hat{x}_n \qquad m = 1, \ldots, N. \tag{39c}$$

If the right-hand-side vector b also has N-th order rotational symmetry, then it is only necessary to solve the single system

$$(A_1 + A_2 + \cdots + A_N)x_1 = b_1 \tag{39d}$$

and set $x_2 = x_3 = \cdots = x_N = x_1$.

For those familiar with Fourier analysis it is easily seen that the symmetry reductions in this section could also be obtained by using discrete Fourier transforms.

2.5 Finite order rotational symmetry with one additional plane of symmetry

Often there is an additional plane of symmetry in problems having finite order rotational symmetry. This case can be handled by successively applying the results of finite order rotational symmetry and those for one plane of symmetry. Assume that each of the rotational blocks A_m of A is further partitioned into two parts A_m^+ and A_m^- consistent with the additional plane of symmetry. Likewise, the blocks of b are partitioned into b_m^+, b_m^- and the blocks of x_m are partitioned into x_m^+, x_m^-. Then the original system of equations can be reduced to the $2N$ smaller systems

$$\left(\sum_{n=1}^{N} e^{i\frac{2\pi(m-1)(n-1)}{N}}\left(A_n^+ + A_n^-\right)\right)\hat{x}_m^+ = \hat{b}_m^+ \qquad m = 1,\ldots,N \tag{40a}$$

$$\left(\sum_{n=1}^{N} e^{i\frac{2\pi(m-1)(n-1)}{N}}\left(A_n^+ - A_n^-\right)\right)\hat{x}_m^- = \hat{b}_m^- \qquad m = 1,\ldots,N \tag{40b}$$

where

$$\hat{b}_m^+ = \frac{1}{2N}\sum_{n=1}^{N} e^{-i\frac{2\pi(m-1)(n-1)}{N}}(b_n^+ + b_n^-) \tag{40c}$$

$$\hat{b}_m^- = \frac{1}{2N}\sum_{n=1}^{N} e^{-i\frac{2\pi(m-1)(n-1)}{N}}(b_n^+ - b_n^-) \tag{40d}$$

and

$$x_m^+ = \sum_{n=1}^{N} e^{i\frac{2\pi(m-1)(n-1)}{N}}(\hat{x}_m^+ + \hat{x}_m^-) \tag{40e}$$

$$x_m^- = \sum_{n=1}^{N} e^{i\frac{2\pi(m-1)(n-1)}{N}}(\hat{x}_m^+ - \hat{x}_m^-). \tag{40f}$$

If the right-hand-side also has the same symmetry, then it is only necessary to solve the single system

$$\left(\sum_{n=1}^{N}\left(A_n^+ + A_n^-\right)\right)x_1^+ = b_1^+ \tag{41}$$

and equate the other blocks x_m^+, x_m^- to x_1^+.

3 Computer Subroutines

The results of the previous section have been implemented in a collection of computer subroutines. These subroutines have been encapsulated in a Fortran 90 module named "symmetry". To use this module the user must insert the command

```
use symmetry
```

at the beginning of their program. In the pages that follow we will describe the various subroutines contained in the module "symmetry" and give their calling syntax.

symsetup

symmetry setup routine

Calling Syntax: call symsetup(nplanes,nrot,irhssym)

nplanes	Number of symmetry planes (0 for none)
nrot	Order of rotational symmetry (0 for none)
irhssym	Flag that is one if the right-hand-side has the same symmetry as the rest of the problem and is zero otherwise.

Purpose:

This routine should be called before any of the other symmetry subroutines. It sets up the symmetry weights and other variables that are used by all the routines.

mtx_combine

performs symmetry combinations on the blocks of input matrix

Calling Syntax: call mtx_combine(A)

A First block row of matrix on which symmetry operations are to be performed, i.e., $A = (A_1, \ldots, A_N)$ where N is the number of symmetry blocks.

Purpose:

This routine separates the input matrix A into blocks A_1,...,A_N and performs the necessary linear combinations of these blocks to produce the reduced size systems to be solved. The reduced matrices are stored over the blocks A_1,...,A_N in the input matrix A.

vec_combine

performs symmetry combinations on the blocks of right-hand-side vector

---

Calling Syntax: call vec_combine(b)

b right-hand-side forcing vector. Only the first block of b is necessary if there is right-hand-side symmetry. The original b is overwritten by the result.

Purpose:

This routine separates the input right-hand-side vector b into blocks and forms the right-hand sides for the various reduced size systems to be solved. The reduced vectors are stored back in the input vector b.

mtxvecmult

performs symmetry reductions on a matrix and a vector and multiplies the resulting reduced matrices and vectors

---

Calling Syntax: call mtxvecmult(B,v,Bv)

B	Block row of input matrix
v	input vector (only first block is needed if there is right-hand-side symmetry)
Bv	Matrix vector product in reduced form.

Purpose:

This routine can be used to form the combined vector of right-hand-sides for the reduced systems arising from systems of equations having the form

$$Ax = Bv$$

where both A and B have the required symmetry and v is given.

decompose

factors blocks of combined input matrix

Calling Syntax: call decompose(A)

A	Block row of input matrix. The input matrix is usually the output matrix from mtx_combine.

Purpose:

This routine is generally called after the symmetry combinations have been performed. It factors the blocks of the matrix A in preparation for solving the smaller reduced systems. A LU-factorization is performed if the system has the same number of equations as unknowns and a householder RU-factorization is performed if there are more equations than unknowns.

solve

solves symmetry reduced systems and combines results to obtain solution of original system

---

Calling Syntax: call solve(A,b,x)

A	block row of decomposed reduced matrices. This matrix is not modified.
b	vector of combined right-hand-side blocks. Only one block is needed if there is right-hand-side symmetry. This vector is modified in the solution process.
x	solution vector. Only the first block is given if there is right-hand-side symmetry.

Purpose:

This routine is called after the reduced matrices have been factored. It solves the reduced systems and combines the solutions to obtain the solution of the original problem. If there are more equations than unknowns, then the systems are solved in the least-square sense.

In order to solve the system $Ax = b$ the routines are called in the following sequence

```
call symsetup(nplanes,nrot,irhssym)
call mtx_combine(A)
call vec_combine(b)
call decompose(A)
call solve(A,b,x).
```

For multiple right-hand sides it is only necessary to repeat the last call for each right-hand-side. If the system to be solved has the form $Ax = Bv$ with v specified, then the calling sequence becomes

```
call symsetup(nplanes,nrot,irhssym)
call mtx_combine(A)
call mtx_combine(B)
call vec_combine(v)
call mtxvecmult(B,v,Bv)
call decompose(A)
call solve(A,Bv,x).
```

4 Computer Source Code

The following pages contain the Fortran 90 source code for the symmetry module. This module contains all of the symmetry subroutines and their shared variables.

```
MODULE symmetry

! Symmetry reduction routines along with shared variables.
save
complex,dimension(:,:),allocatable :: symwts
complex,dimension(:),allocatable :: rootun
integer :: nblks,irhssym=0,isymtype
integer,dimension(:,:),allocatable :: ipvt
complex,dimension(:,:),allocatable :: qraux
complex,pointer,dimension(:,:) :: MtxPtr
complex,pointer,dimension(:) :: VecPtr
real,dimension(8,8) :: refwts

CONTAINS

SUBROUTINE symsetup(nplanes,nrot,irhssymmetry)

! nplanes=number of planes of symmetry
! nrot=order of rotational symmetry
! irhssymmetry=1 if right-hand-side forcing vector has same symmetry as rest of problem
! This routine determines type of symmetry, the number of symmetry blocks, and
! calculates symmetry weights.
! irhssym=1 if right-hand-side of equations has same symmetry as matrix.
! isymtype=1 for reflective symmetry
! isymtype=2 for finite order rotational symmetry
! isymtype=3 for rotational plus an additional plane of symmetry

refwts=reshape((/ 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,&
                  1.0,-1.0, 1.0,-1.0, 1.0,-1.0, 1.0,-1.0,&
                  1.0, 1.0,-1.0,-1.0, 1.0, 1.0,-1.0,-1.0,&
                  1.0,-1.0,-1.0, 1.0, 1.0,-1.0,-1.0, 1.0,&
                  1.0, 1.0, 1.0, 1.0,-1.0,-1.0,-1.0,-1.0,&
                  1.0,-1.0, 1.0,-1.0,-1.0, 1.0,-1.0, 1.0,&
                  1.0, 1.0,-1.0,-1.0,-1.0,-1.0, 1.0, 1.0,&
                  1.0,-1.0,-1.0, 1.0,-1.0, 1.0, 1.0,-1.0/),(/8,8/))
irhssym=irhssymmetry
pi=acos(-1.0)
if(nplanes/=0 .and. nrot==0) then
  isymtype=1    ! reflective symmetry
  nblks=2**nplanes
end if
if(nplanes==0 .and. nrot/=0) then
  isymtype=2    ! finite order rotational symmetry
  nblks=nrot
end if
if(nplanes/=0 .and. nrot/=0) then
  if(nplanes /= 1) then
    write(*,*) 'There can only be one plane of symmetry along with rotational symmetry'
    stop
  end if
  isymtype=3    ! finite order rotational symmetry plus one symmetry plane
  nblks=nrot
end if
allocate(symwts(nblks,nblks))    ! allocate symmetry weight matrix
```

```
if(isymtype==1) symwts=cmplx(refwts(1:nblks,1:nblks),0.)  ! reflective symmetry weights
if(isymtype==2 .or. isymtype==3) then
  fact=2.0*pi/nblks
  allocate(rootun(nblks))
  do i=1,nblks
    rootun(i)=cmplx(cos((i-1)*fact),sin((i-1)*fact))   ! roots of unity
  end do
  do i=1,nblks
    do j=1,nblks
      k=mod((i-1)*(j-1),nblks)
      symwts(i,j)=rootun(k+1)   ! rotational symmetry weights
    end do
  end do
end if
return
END SUBROUTINE symsetup

!**********************************************************************

SUBROUTINE mtx_combine(A)

! A contains a block row of system matrix.
! This routine performs symmetry combinations on blocks of matrix A in order to obtain
! matrices for reduced systems. The matrices for reduced systems are stored over blocks of A.

complex,dimension(:,:),target :: A
complex, dimension(size(A,1),size(A,2)/(nblks**irhssym)),target :: work

if(mod(size(A,2),nblks) /= 0) then
  write(*,*) 'Number of columns of input matrix is not divisible by the number of symmetry &
  &blocks in subroutine mtx_combine'
  stop
end if
nrows=size(A,1)
ncols=size(A,2)/nblks
work=(0.0,0.0)
nblks1=nblks
if(irhssym==1) nblks1=1
do i=1,nblks1
MtxPtr => work(1:nrows,1+(i-1)*ncols:i*ncols)
  do j=1,nblks
    MtxPtr=MtxPtr+symwts(i,j)*A(1:nrows,1+(j-1)*ncols:j*ncols) ! symmetry combination of blocks
  end do
end do
if(isymtype==3) then
  do i=1,nblks1    ! additional combinations for extra plane of symmetry
    A(1:nrows,1+(i-1)*ncols:i*ncols-ncols/2)=work(1:nrows,1+(i-1)*ncols:i*ncols-ncols/2)+&
    work(1:nrows,i*ncols-ncols/2+1:i*ncols)
    A(1:nrows,i*ncols-ncols/2+1:i*ncols)=work(1:nrows,1+(i-1)*ncols:i*ncols-ncols/2)-&
    work(1:nrows,i*ncols-ncols/2+1:i*ncols)
  end do
else
  A=work
end if
```

```
return
END SUBROUTINE mtx_combine

!***********************************************************************

SUBROUTINE vec_combine(b)

! This routine performs symmetry combinations on blocks of right-hand-side vector b
! in order to obtain right-hand-sides of the reduced systems. The reduced right-hand-sides
! are stored over the original blocks of b.

complex,dimension(:),target :: b
complex, dimension(size(b)/(nblks**irhssym)),target :: work

if(irhssym==1) return
if(mod(size(b),nblks) /= 0) then
  write(*,*) 'Number of components of input vector is not divisible by the number of symmetry &
  &blocks in subroutine vec_combine'
  stop
end if
nrows=size(b)/nblks
work=(0.0,0.0)
do i=1,nblks
VecPtr => work(1+(i-1)*nrows:i*nrows)
  do j=1,nblks
    VecPtr=VecPtr+conjg(symwts(i,j))*b(1+(j-1)*nrows:j*nrows)  ! symmetry combination of vectors
  end do
end do
b=work/nblks
if(isymtype==3) then
  work=b
  do i=1,nblks  ! additional combination for extra plane of symmetry
    b(1+(i-1)*nrows:i*nrows-nrows/2)=0.5*(work(1+(i-1)*nrows:i*nrows-nrows/2)+&
    work(i*nrows-nrows/2+1:i*nrows))
    b(i*nrows-nrows/2+1:i*nrows)=0.5*(work(1+(i-1)*nrows:i*nrows-nrows/2)-&
    work(i*nrows-nrows/2+1:i*nrows))
  end do
end if
return
END SUBROUTINE vec_combine

!***********************************************************************

SUBROUTINE mtxvecmult(B,v,Bv)

! This routine forms the right-hand-sides for the reduced systems when the original system
! has the form Ax=Bv with v given. The reduced right-hand-sides are stored in Bv.

complex,dimension(:,:),target :: B
complex,dimension(:),target :: v
complex,dimension(:):: Bv

nb=nblks
if(isymtype==3) nb=2*nb
```

```
if(mod(size(B,2),nb) /= 0) then
  write(*,*) 'Number of columns of input matrix is not divisible by the number of symmetry &
  &blocks in subroutine mtxvecmult'
  stop
end if
if(size(v) < size(B,2)) then
  write(*,*) 'Input vector must be at least as large as the number of columns in the &
  &input matrix in subroutine mtxvecmult'
  stop
end if
nrows=size(B,1)
ncols=size(B,2)/nb
if(irhssym==1) then
  Bv(1:nrows)=matmul(B(1:nrows,1:ncols),v(1:ncols))
  return
end if

do i=1,nb
MtxPtr => B(1:nrows,1+(i-1)*ncols:i*ncols)
VecPtr => v(1+(i-1)*ncols:i*ncols)
  Bv(1+(i-1)*nrows:i*nrows)=matmul(MtxPtr,VecPtr)
end do
return
END SUBROUTINE mtxvecmult

!*****************************************************************************

SUBROUTINE decompose(A)

! This routine factors the blocks of the combined block row A that is produced by
! mtx_combine. If the blocks are square a LU decomposition is performed. If the blocks are
! overdetermined, a QR factorization is performed. The factored matrices are stored back in A.

complex,dimension(:,:) :: A
complex,dimension(size(A,1),size(A,1)) :: Atemp
integer,dimension(size(A,1)) :: ipvti
complex,dimension(size(A,1)) :: qrauxi
complex,dimension(size(A,1)) :: work
nrows=size(A,1)
nb=nblks
if(isymtype==3) nb=2*nb
if(mod(size(A,2),nb) /= 0) then
  write(*,*) 'Number of columns of input matrix is not divisible by the number of symmetry &
  &blocks in subroutine decompose'
  stop
end if
ncols=size(A,2)/nb
nb1=nb
if(irhssym==1) nb1=1
allocate(ipvt(ncols,nb))
if(nrows==ncols) then
  do i=1,nb1
    Atemp(1:nrows,1:ncols)=A(1:nrows,1+(i-1)*ncols:i*ncols)
    call cgefa(Atemp,nrows,nrows,ipvti,info)
```

```
    A(1:nrows,1+(i-1)*ncols:i*ncols)=Atemp(1:nrows,1:ncols)
    ipvt(:,i)=ipvti(1:ncols)
  end do
else
  do i=1,nb1
    Atemp(1:nrows,1:ncols)=A(1:nrows,1+(i-1)*ncols:i*ncols)
    call cqrdc(Atemp,nrows,nrows,ncols,qrauxi,ipvti,work,1)
    A(1:nrows,1+(i-1)*ncols:i*ncols)=Atemp(1:nrows,1:ncols)
    ipvt(:,i)=ipvti(1:ncols)
    qraux(:,i)=qrauxi(1:ncols)
  end do
end if
return
END SUBROUTINE decompose

!*********************************************************************

SUBROUTINE solve(A,b,x)

! This routine solves the reduced systems whose decomposed blocks are stored in A and whose
! right-hand-sides are stored in b. The solutions of the reduced systems are then combined
! to obtain the solution of the original system of equations. This solution is output in x.

complex,dimension(:,:) :: A
complex,dimension(:),target :: b,x
complex,dimension(size(A,1),size(A,1)) :: Atemp
complex,dimension(size(A,1)) :: btemp,dummy,soln
integer,dimension(size(A,1)) :: ipvti
complex,dimension(size(A,1)) :: qrauxi
complex,dimension(2*size(A,1)) :: work

nrows=size(A,1)
nb=nblks
if(isymtype==3) nb=2*nb
if(mod(size(A,2),nb) /= 0) then
  write(*,*) 'Number of columns of input matrix is not divisible by the number of symmetry &
  &blocks in subroutine solve'
  stop
end if
if(size(b) < size(A,1)) then
  write(*,*) 'Right-hand-side vector must be at least as large as the number of rows in input matrix'
  stop
end if
if(size(x) < size(A,1)) then
  write(*,*) 'Solution vector must be at least as large as the number of rows in input matrix'
  stop
end if
ncols=size(A,2)/nb
nb1=nb
if(irhssym==1) nb1=1
if(nrows==ncols) then
  do i=1,nb1
    Atemp(1:nrows,1:ncols)=A(1:nrows,1+(i-1)*ncols:i*ncols)
    ipvti(1:ncols)=ipvt(:,i)
```

```
    btemp(1:nrows)=b(1+(i-1)*nrows:i*nrows)
    call cgesl(Atemp,nrows,nrows,ipvti,btemp,0)
    x(1+(i-1)*ncols:i*ncols)=btemp(1:ncols)
  end do
else
  do i=1,nb1
    Atemp(1:nrows,1:ncols)=A(1:nrows,1+(i-1)*ncols:i*ncols)
    qrauxi(1:ncols)=qraux(:,i)
    call cqrsl(Atemp,nrows,nrows,ncols,qrauxi,btemp,dummy,btemp,soln,dummy,dummy,100,info)
    x(1+(i-1)*ncols:i*ncols)=soln(ipvt(:,i))
  end do
end if
if(irhssym==1) return
if(isymtype==3) then
  do i=1,nblks
    work(1:2*ncols)=x(1+(i-1)*2*ncols:i*2*ncols)
    x(1+(i-1)*2*ncols:(2*i-1)*ncols)=work(1:ncols)+work(ncols+1:2*ncols)
    x((2*i-1)*ncols+1:i*2*ncols)=work(1:ncols)-work(ncols+1:2*ncols)
  end do
  ncols=2*ncols
end if
b(1:size(x))=x
x=(0.,0.)
do i=1,nblks
  VecPtr => x(1+(i-1)*ncols:i*ncols)
  do j=1,nblks
    VecPtr=VecPtr+symwts(i,j)*b(1+(j-1)*ncols:j*ncols)
  end do
end do
return
END SUBROUTINE solve

END MODULE symmetry
```