USING SPARSE MATRIX TECHNIQUES TO SOLVE A MODEL OF THE WORLD ECONOMY

D.B. Szyld

(Institute for Economic Analysis, New York University)

1. INTRODUCTION

In this paper we review some sparse matrix techniques which lead to a very efficient algorithm for the solution of a model of the world economy.

We give a short mathematical description of the model, from which we can study its underlying structure, and thereby choose the appropriate method to solve its equations, a suitable data structure, and an algorithm that can utilise high quality software.

2. STATEMENT OF THE PROBLEM

The Input-Output model of the world economy (IO model) developed at the Institute for Economic Analysis divides the world into r regions ($r \sim 15$). For each of the regions there is a set of linear algebraic equations of the form

$$A_i \underline{y}_i + S_i \underline{w} = \underline{0} \qquad (i = 1,\ldots,r). \tag{1}$$

In addition there are global equations

$$\sum_{i=1}^{r} G_i \underline{y}_i = \underline{0}. \tag{2}$$

A more detailed description of the IO model can be found in Duchin and Szyld (1979), Petri (1977), and Leontief, Carter and Petri (1977).

All the matrices involved are very sparse. For example

A_i could be 200 × 250 with 2500 non-zeros.

S_i could be 200 × 50 with 50 non-zeros.

G_i could be 50 × 250 with 100 non-zeros.

Each matrix A_i has more columns than rows and therefore some components of $\underline{y}_i$ have to be prescribed.

Denote by $\underline{z}_i$ the prescribed components of $\underline{y}_i$, and the unknowns by $\underline{x}_i$, and partition A_i and G_i accordingly; then we can rewrite (1) and (2) as

$$M_i \underline{x}_i + S_i \underline{w} = -B_i \underline{z}_i = \underline{b}_i \quad (i=1,\ldots,r). \tag{3}$$

$$\sum_{i=1}^{r} E_i \underline{x}_i = -\sum_{i=1}^{r} E'_i \underline{z}_i = \underline{b}_{r+1}. \tag{4}$$

In fact the structure of the matrices G_i guarantees that $\underline{b}_{r+1} = \underline{0}$, but this fact is not exploited in the implementation design.

The entire model is thus a linear system of equations with a nonsymmetric bordered block diagonal matrix of coefficients of the form:

$$\begin{pmatrix} M_1 & & & & S_1 \\ & M_2 & & & S_2 \\ & & \ddots & & \vdots \\ & & & M_r & S_r \\ E_1 & E_2 & \cdots & E_r & O \end{pmatrix} \begin{pmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \vdots \\ \underline{x}_r \\ \underline{w} \end{pmatrix} = \begin{pmatrix} \underline{b}_1 \\ \underline{b}_2 \\ \vdots \\ \underline{b}_r \\ \underline{0} \end{pmatrix} \tag{5}$$

where the blank blocks in the matrix are zero blocks.

Our aim was to solve the system exploiting its block structure and the sparsity in each block, using state of the art techniques, to design a flexible algorithm to handle any r and M_i of different sizes, and to use a storage structure compatible with existing software and which is efficient for data management.

In contrast the Institute had a code that stored the given matrices in packed form and computed and stored all the inverses M_i^{-1}. The approximate computer time was 4 hours on a PDP-11. The (dense) inverses were saved for subsequent runs. When designing the solution algorithm we had in mind the size of the matrices, and that each matrix M_i is non singular, making Block

Gaussian Elimination suitable. Some research was performed on similar problems using iterative methods and we favoured a direct method approach and the use of available high quality software.

3. REVIEW OF BLOCK GAUSSIAN ELIMINATION

Block Gaussian Elimination goes back at least to Householder (1964) and has special advantages in sparse cases. Given a linear system

$$A\underline{x} = \underline{b},$$

we can partition the matrix as:

$$A = \begin{pmatrix} A_{11} & S \\ E & A_{22} \end{pmatrix}.$$

If we factor $A_{11} = L_1 U_1$, then we have

$$A = \begin{pmatrix} L_1 & O \\ EU_1^{-1} & I \end{pmatrix} \begin{pmatrix} U_1 & L_1^{-1}S \\ O & C \end{pmatrix}$$

where C is the so-called Schur complement

$$C = A_{22} - EA_{11}^{-1}S = A_{22} - EU_1^{-1}L_1^{-1}S.$$

In fact, the off diagonal blocks EU_1^{-1} and $L_1^{-1}S$ do not have to be stored; their columns are computed one at a time, used and discarded.

We can now rewrite $A\underline{x} = \underline{b}$ as:

$$\begin{pmatrix} L_1 & O \\ EU_1^{-1} & I \end{pmatrix} \begin{pmatrix} U_1 & L_1^{-1}S \\ O & C \end{pmatrix} \begin{pmatrix} \underline{x}_1 \\ \underline{x}_2 \end{pmatrix} = \begin{pmatrix} \underline{b}_1 \\ \underline{b}_2 \end{pmatrix}$$

The solution $\underline{x}$ is obtained by solving first

$$\begin{pmatrix} L_1 & 0 \\ EU_1^{-1} & I \end{pmatrix} \begin{pmatrix} \underline{g}_1 \\ \underline{g}_2 \end{pmatrix} = \begin{pmatrix} \underline{b}_1 \\ \underline{b}_2 \end{pmatrix}$$

i.e.,

(1) obtained $\underline{g}_1$ from $L_1\underline{g}_1 = \underline{b}_1$

(2) obtain $\underline{e}$ from $U_1\underline{e} = \underline{g}_1$

(3) obtain $\underline{g}_2$ from $E\underline{e} + \underline{g}_2 = \underline{b}_2$

and then solving

$$\begin{pmatrix} U_1 & L_1^{-1}S \\ 0 & C \end{pmatrix} \begin{pmatrix} \underline{x}_1 \\ \underline{x}_2 \end{pmatrix} = \begin{pmatrix} \underline{g}_1 \\ \underline{g}_2 \end{pmatrix}$$

i.e.,

(1) solve $C\underline{x}_2 = \underline{g}_2$

(2) solve $L_1\underline{e}' = S\underline{x}_2$

(3) solve $U_1\underline{x}_1 + \underline{e}' = \underline{g}_1$.

Thus, the off diagonal blocks are not needed explicitly for the solution but only when computing the Schur complement C: denote any column of C by $\underline{c}$, the corresponding column of S by $\underline{s}$ and that of A_{22} by $\underline{a}$.

Solve $$A_{11}\underline{u} = L_1U_1\underline{u} = \underline{s}$$

and compute $$\underline{c} = \underline{a} - E\underline{u}$$

The step $$C\underline{x}_2 = \underline{g}_2$$

can be done by factoring $C = L_{22}U_{22}$, or by partitioning C and using Block Gaussian Elimination recursively. Further

application of these ideas can be found in George (1980). In the first case we have,

$$A = \begin{pmatrix} L_1 & O \\ EU_1^{-1} & L_{22} \end{pmatrix} \begin{pmatrix} U_1 & L_1^{-1}S \\ O & U_{22} \end{pmatrix}$$

We have then recovered the LU factors of A. The advantage of our approach is that we only store the factors of A_{11} and C, and that fill-in occurs in these diagonal blocks.

Typically C is full, and Block Gaussian Elimination is advantageous specially when the order of C is small compared with that of A.

Note that if A is block diagonal, that is E = O and S = O, its LU factors are block diagonal, with its blocks being the factors of the diagonal blocks of A. Therefore, if A_{11} is block diagonal, L_1 is block diagonal and so is U_1.

We now consider our special problem (see section 2). Since A_{11} is block diagonal, its diagonal blocks being the matrices M_i, we can write the Schur complement

$$C = EA_{11}^{-1}S = \sum_{i=1}^{r} E_i M_i^{-1} S_i$$

We use the subroutines from MA28 (Duff (1977)) to solve the systems for each matrix M_i. This code looks for a block triangular form for each M_i, and factors its diagonal blocks. The Schur complement is then computed by accumulating it column by column as explained before. We use a square array to store C, since it is expected to be full, and factor it using suitable LINPACK subroutines (Dongarra et al (1979)). The complete solution algorithm is given by:

1. For i = 1,...,r

   1.1 Obtain M_i and $\underset{-i}{b} = B_i \underset{i-i}{z}$

   2.1 Obtain the factors of M_i

2. For $i = 1,\ldots,r$

   2.1 For each column of S_i and for $\underline{b}_i$,

      2.1.1 Assign that column or vector to $\underline{v}$

      2.1.2 Solve $M_i \underline{u} = \underline{v}$

      2.1.3 Compute $E_i \underline{u}$ and subtract it from the corresponding column of E_{r+1} or $\underline{b}_{r+1}$.
      (E_{r+1} and $\underline{b}_{r+1}$ are zero initially)

The resulting matrix is the Schur complement C and the resulting vector is called $\underline{f}$.

3. Factor C obtaining L_{22} and U_{22}

4. Solve $C\underline{w} = \underline{f}$

5. For $i = 1,\ldots r$

   5.1 Compute $\hat{\underline{b}}_i = \underline{b}_i - S_i \underline{w}$

   5.2 Solve $M_i \underline{x}_i = \hat{\underline{b}}_i$

4. DATA STRUCTURES

Each matrix A_i (the columns of which generate M_i and B_i) is stored in compressed columnwise form, as described by Gustavson (1978). We show how the non-zero elements are being handled, while the row indices and pointers are treated in a similar way.

A large array ABIG is used as storage for the non-zero entries of the matrices A_i, M_i and its factors. The location in ABIG of the first non-zero element of A_i is stored in the i-th position of a short array IDIVA.

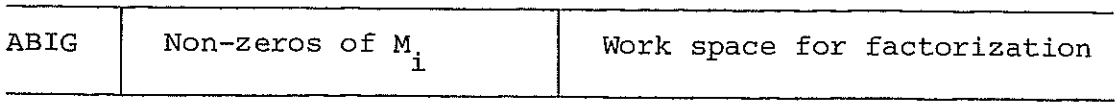
We show in detail step 1 of the algorithm

For i = 1,...r

- Set IDIVA (i) (where IDIVA (1) = 1)
- Read the non-zeros of A_i into core.

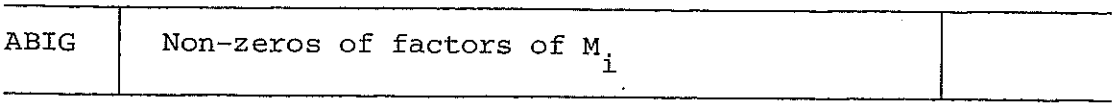
IDIVA (i)

- Read a set of integers defining B_i, and read z_{-i}.
- Accumulate $b_{-i} = B_i z_{-i}$.
- The remaining columns of A_i correspond to M_i.
  Compress the non-zeros of A_i to obtain M_i.



IDIVA (i)

- Obtain factorization of M_i (Analyze-Factor step of MA28)



IDIVA (i) IDIVA (i+1)

5. EXPERIMENTAL CONCLUSIONS AND FURTHER REMARKS

In our runs, we have observed about 10% fill-in in each of the matrices M_i.

Runs on a IBM 370/168 required about 150 seconds CPU time for 4 decades, that is for 4 sets of linear systems, using double precision and 1 Megabyte of storage.

If 1 Megabyte of storage is not available, the factors of the matrices M_i could be stored out of core and be recalled when needed.

We shall now explore the feasibility of a "hybrid" method. Namely, the use of an iterative method for each diagonal block M_i, but we still compute the Schur complement and use Block Gaussian Elimination as described before.

For a typical matrix M_i the analyze and factor step takes between 300 and 455 milliseconds CPU time, while the solve step takes about 11 milliseconds CPU time. Because of the small amount of fill-in observed, the time for backsolving is comparable to the time of a matrix vector multiplication, and therefore comparable to the time of a single iteration in an iterative method with M_i as the matrix involved in the iteration. The time for the analyze and factor step is comparable with the time for about 27 to 42 iterations.

In our algorithm we have to solve about 50 systems of the form $M_i \underline{v} = \underline{f}$ for each matrix M_i (steps 2.1.2 and 5.2). The convergence properties of an iterative method for $M_i \underline{v} = \underline{f}$ have not been studied, but in any case an iterative method will be superior only if it converges in 2 iterations to an accurate solution. Thus, the "hybrid" approach of using an iterative method for each diagonal block is not competitive with our algorithm.

To study the feasibility of an iterative method involving the whole matrix we should bear in mind that computing the Schur complement requires as much time as the factorization of all the diagonal blocks M_i. Therefore, the time involved in our solution algorithm is comparable to 50 to 80 iterations using the entire matrix of the system (5). Yet it is not clear that there is a reliable iterative method for this particular problem, that is for a system with a matrix of the structure described.

REFERENCES

Dongarra, J.J., Bunch, J.R., Moler, C.B., and Stewart, G.W. (1979) LINPACK, User's Guide, SIAM Press.

Duchin, F. and Szyld, D.B. (1979) Application of sparse matrix techniques to inter-regional input-output analysis, Economics of Planning, Vol. 15 , in press.

Duff, I.S. (1977) MA28 - a Set of FORTRAN Subroutines for Sparse Unsymmetric Systems of Linear Equations, Report R. 8730, A.E.R.E. Harwell, HMSO, London.

George, A. (1980) Direct solution of sparse positive definite systems: some basic ideas and open problems, These proceedings.

Gustavson, F.G. (1978) Two fast algorithms for sparse matrices: multiplication and permuted transposition, TOMS 4, pp. 250-269.

Householder, A.S. (1964) The Theory of Matrices in Numerical Analysis, Blaidsell.

Leontief, W., Carter, A., and Petri, P. (1977) The Future of the World Economy, Oxford University Press.

Petri, P. (1977) Guide to the operation of the United Nations world model, Technical Report No. 3, Department of Economics and Brandeis Economics Research Center, Brandeis University.

Sparse Matrices and their Uses

Based on the proceedings of the IMA Numerical Analysis Group Conference, organised by The Institute of Mathematics and its Applications and held at the University of Reading, 9th-11th July, 1980

Edited by

IAIN S. DUFF

AERE Harwell
Didcot, Oxfordshire
England.

1981

ACADEMIC PRESS

A Subsidiary of Harcourt Brace Jovanovich, Publishers

London New York Toronto Sydney San Francisco