March 29 - April 7, 2009
St.Petersburg, Russia
Joint Advanced Student School (JASS) 2009

Methods of solving sparse linear systems

Oleg Soldatenko
St.Petersburg State University
Faculty of Physics
Department of Computational Physics

Introduction

A system of linear equations is called sparse if only relatively small number of its matrix elements are nonzero. It is wasteful to use general methods of linear algebra for such problems because when we use all the elements of matrix (zero and nonzero) we perform N^3 operations. For example if we use Gauss method. There are many types of sparse matrices and special methods of their solution. For example I want to show you some general types of matrices:

Tridiagonal, band diagonal with bandwidth M, block tridiagonal etc.

But we are often faced with matrices which are similar to these types, but are not exactly equivalent to them. Therefore we need to find a method which we could use for such slightly perturbed matrices. For these one can use Sherman-Morrison formula.

Sherman-Morrison formula

The general idea of Sherman-Morrison formula is in the representation of the original matrix as the sum matrix A for which inversion can be performed and a tensor product of vectors u and v.

$$\boldsymbol{A} \longrightarrow (\boldsymbol{A} + \boldsymbol{u} \otimes \boldsymbol{v})$$

The Sherman-Morrison formula can be directly applied to a class of sparse problems. If you already have a fast way of calculating the inverse of $\boldsymbol{A}$, then this method allows you to build up a method for more complicated matrices, adding for example a row or a column at a time. Notice that you can apply the Sherman-Morrison formula more than once successively, using at each stage the most recent update of $\boldsymbol{A}^{-1}$. Of course, if you have to modify every row, then you are back the an $\boldsymbol{N}^3$ method.

For some other sparse problems, the Sharman-Morrison formula cannot be directly applied for the simple reason that storage of the whole inverse matrix $\boldsymbol{A}^{-1}$ is not feasible. If you want to add only a single correction of the form $\boldsymbol{u} \otimes \boldsymbol{v}$, and solve the linear system

$$(\boldsymbol{A} + \boldsymbol{u} \otimes \boldsymbol{v}) \cdot \boldsymbol{x} = \boldsymbol{b}$$

Then you proceed as follows. Using the fast method that is presumed available for the matrix $\boldsymbol{A}$, solve the two auxiliary problems

$$A \cdot y = b \qquad\qquad A \cdot z = u$$

For the vectors y and z. In terms of these solutions

$$x = y - \left(\frac{v \cdot y}{1 + (v \cdot z)}\right) z$$

Example of cyclic tridiagonal system

I consider cyclic tridiagonal system. The equations have the form:

$$\begin{bmatrix} b_0 & c_0 & 0 & \cdots & & & \beta \\ a_1 & b_1 & c_1 & \cdots & & & \\ & & & \cdots & a_{N-2} & b_{N-2} & c_{N-2} \\ \alpha & & & \cdots & 0 & a_{N-1} & b_{N-1} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \\ \cdots \\ r_{N-2} \\ r_{N-1} \end{bmatrix}$$

This is a tridiagonal system, except for the matrix elements α and β in the corners.

We use the Sherman-Morrison formula, treating the system as a tridiagonal plus a correction. Vectors u and v are

$$u = \begin{bmatrix} \gamma \\ 0 \\ \vdots \\ 0 \\ \alpha \end{bmatrix} \qquad\qquad v = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \beta/\gamma \end{bmatrix}$$

here γ is arbitrary for the moment. Then the matrix A is the tridiagonal part of initial matrix, with two terms modified:

$$b'_0 = b_0 - \gamma, \qquad\qquad b'_{N-1} = b_{N-1} - \frac{\alpha\beta}{\gamma}$$

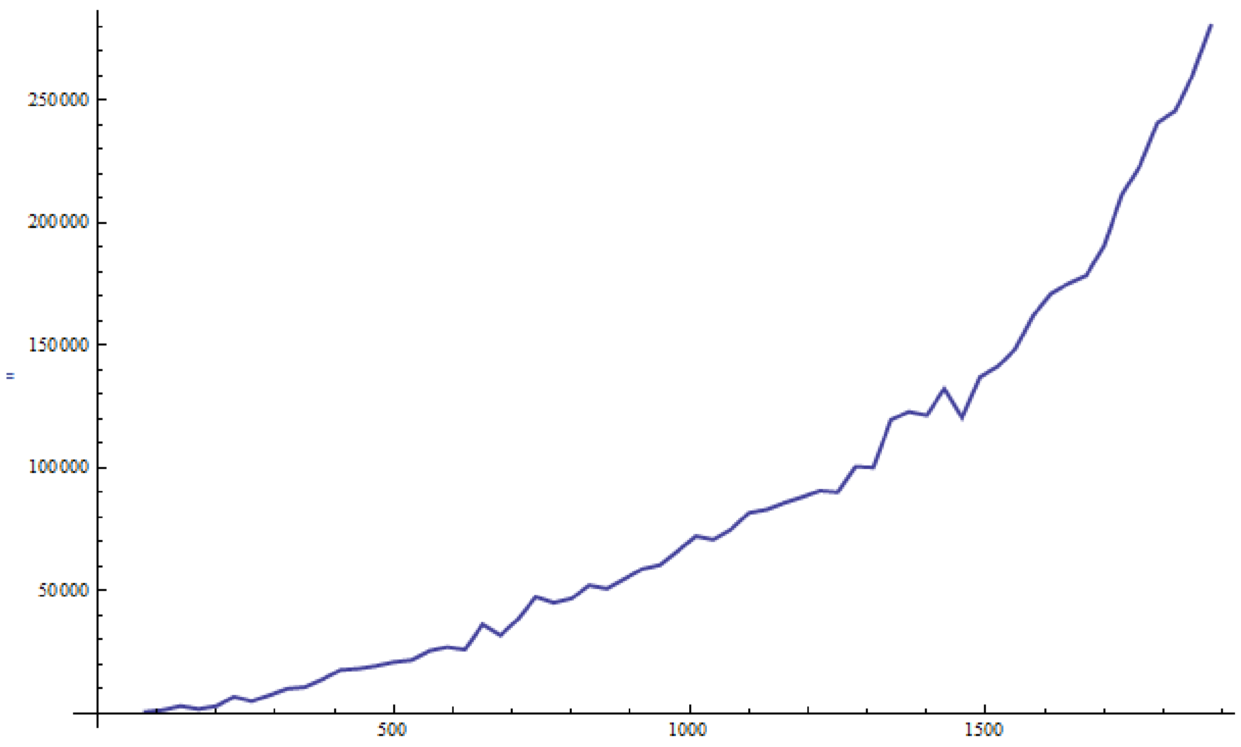
We solve equations

$$A \cdot y = b \qquad\qquad A \cdot z = u$$

With the standard tridiagonal algorithm, for example sweep method, and then get the solution by the formula

$$x = y - \left(\frac{v \cdot y}{1 + (v \cdot z)}\right) z$$

I have developed the program that solves this cyclic system and obtained results, which are presented on the picture.



The abscissa shows the dimension of the matrix, and the Y-axis is the ratio of the time for Gauss method to the time of Sharman-Morrison algorithm. In this graphic we see, that acceleration of the method is proportional to N^2. The same result is predicted by the theory.

Woodbury Formula

If you want to add more than a single correction term, then you cannot use Sherman-Morrison formula repeatedly, since without storing a new A^{-1} you will not be able to solve the auxiliary problems efficiently after the first step. Instead you need the Woodbury Formula,

$$A \longrightarrow (A + U \cdot V^T)$$

Here A is, as usual, an $N \times N$ matrix, while U and V are $N \times P$ matrices with $P < N$ and usually $P \ll N$.

The relation between the Woodbury formula and successive applications of the Sherman-Morrison formula is now clarified by noting that, if U is the matrix formed by columns with P vectors $u_0, \ldots, u_{P-1}$, and V is the matrix formed by columns with P vectors $v_0, \ldots, v_{P-1}$.

Then two ways of expressing the same correction to A are

$$\left(A + \sum_{k=0}^{P-1} u_k \otimes v_k \right) = (A + U \cdot V^T)$$

Note that the subscripts on **u** and **v** do not denote components, but rather distinguish the different column vectors.

Last equation reveals that, if you have A^{-1} in storage, then you can either make the P corrections at once fell swoop by using Woodbury formula, inverting a $P \times P$ matrix, or else make them by applying Sherman-Morrison formula P successively.

If you don't have storage for A^{-1}, then you must use Woodbury formula in the following way: To solve the linear equation

$$\left(A + \sum_{k=0}^{P-1} u_k \otimes v_k\right) \cdot x = b$$

first solve the P auxiliary problems

$$\begin{gathered} A \cdot z_0 = u_0 \\ A \cdot z_1 = u_1 \\ \dots \\ A \cdot z_{P-1} = u_{P-1} \end{gathered}$$

and construct the matrix Z by columns of the obtained solutions

$$Z \equiv \left[\begin{bmatrix} \\ z_0 \\ \\ \end{bmatrix} \dots \begin{bmatrix} \\ z_{P-1} \\ \\ \end{bmatrix}\right]$$

Next, do the $P \times P$ matrix inversion

$$H \equiv (1 + V^T \cdot Z)^{-1}$$

Finally, solve one more auxiliary problem

$$A \cdot y = b$$

In terms of these quantities, the solution is given by the formula

$$x = y - Z \cdot [H \cdot (V^T \cdot y)]$$

This algorithm is applicable to a wider class of matrices, than Sherman-Morrison algorithm.

Indexed Storage of Sparse Matrices

I would like to discuss another problem connected with sparse matrices, it is indexig of matrix elements. It is prodigal to store all the elements of a matrix if there is a lot of zeros. Therefore I would present a method which allows to store only two vectors which length is equal to the number of nonzero elements of a matrix. Assume that we have a matrix.

$$A = \begin{bmatrix} 3 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 7 & 5 & 9 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 6 & 5 \end{bmatrix}$$

Index k		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ija[k]		**6**	**7**	**7**	**9**	**10**	**11**	**2**	**1**	**3**	**4**	**2**
sa[k]		**3**	**4**	**5**	**0**	**5**	**x**	**1**	**7**	**9**	**2**	**6**

The first N cells of **sa** store diagonal matrix elements of **A**, in their order. (Note that diagonal elements are stored even if they are zero; this is at most a slight storage inefficiency, since diagonal elements are nonzero in most realistic applications)
Each of the first N cells of **ija** stores the index of the array **sa** that contains the first off-diagonal element of the corresponding row of the matrix. (If there are no off-diagonal elements for that row, it is by one greater than the index in **sa** of the most recently stored elements of a previous row)

The cell 0 of **ija** is always equal to N +1. (It can be read to determine N)

The cell N of **ija** is by one larger than the index in **sa** of the last off-diagonal element of the last row. (It can be read to determine the number of nonzero elements in the matrix, or the number of elements in the arrays **sa** and **ija**.) The cell N of **sa** is not used and can be set arbitrarily.

Entries in **sa** at cells >= N+1 contain A's off-diagonal values, ordered by rows and, within each row, ordered by columns.

Entries in **ija** at cells >= N+1 contain the column index of the corresponding element in **sa**.

Conclusions

- Have been presented Sherman-Morrison and Woodbury methods to solving sparse linear systems which are faster in comparison with standard methods in some cases.
- Has been presented the method of storage of a sparse matrix which allows to spend less memories.

References

- W.H.Press, S.A. Teukolsky, W. T. Wetterling, B.P Flannery

"Numerical Recipes in C++",Cambridge University Press, 2002.