Automatic Generation of Compatibility Conditions for a Structural Beam

A technique for automating the computationally intensive task of generating compatibility conditions for a structure like a **Beam**.

Saurabh V. Pendse

The Maharaja Sayajirao University of Vadodara

January 19, 2010

Abstract

The main aim of this project is to solve a part of a much larger problem. The larger problem deals with the analysis of non-linear structures using a method known as the *Integrated Force Method*. One of the key components required for the analysis of structures using this method is the *Compatibility Matrix*, whose creation is a non-trivial and computationally intensive process. The main aim of this project is to develop a reliable and efficient method to automate the task of generating the Compatibility Matrix for a given structure, so as to aid in the further analysis of the structure.

Outline

Notations Used

We use

- **B** - to describe the **Equilibrium Matrix**.
- **F** - to describe the **Internal Force Matrix**.
- **V** - to describe the **External Load Matrix**.
- **G** - to describe the **Flexibility Matrix**.
- **N** - to describe the **Nodal Displacement Matrix**.
- **I** - to describe the **Internal Energy of the Structure**
- β - to describe the **Generalized Internal Deformations Vector**.

Basic Theory of the Integrated Force Method I

The Integrated Force Method is one of the standard methods for the analysis of any discrete structure. According to this method, any structure can be described through two attributes i.e. we may define a structure in the form of **struct(n,m)**, where

- *n* - Forces
- *m* - Degrees of Freedom of the Structure

The basic equation of the Integrated Force Method is as follows. For any discrete structure, the following equation holds true: The matrix **B** is sparse and rectangular in nature, and the process of generating the B matrix for a given structure is quite trivial. The matrix **F** is the vector sum of all the internal forces of the structure. The matrix **V** represents the external load applied to the structure.

$$[\mathbf{B}]\{\mathbf{F}\} = \{\mathbf{V}\} \tag{1.1}$$

Basic Theory of the Integrated Force Method II

The internal energy of the structure considering the nodal displacements $\{\mathbf{N}\}$ of the structure can be written as:

$$IE = \frac{1}{2}\{\mathbf{N}\}^T\{\mathbf{V}\} = \frac{1}{2}\{\mathbf{N}\}^T[\mathbf{B}]\{\mathbf{F}\} \tag{1.2}$$

The internal energy of the structure can also be expressed considering the deformations of the elements as follows:

$$IE = \frac{1}{2}\{\mathbf{F}\}^T\{\beta\} \tag{1.3}$$

where $\{\beta\}$ represents the vector of generalized internal deformations of the elements.
Subtracting (1.3) from (1.2), we can write

$$\frac{1}{2}\{\mathbf{F}\}^T([\mathbf{B}]^T\{\mathbf{N}\} - \{\beta\}) = 0 \tag{1.4}$$

Basic Theory of the Integrated Force Method III

As $\{\mathbf{F}\}$is not null, we have

$$[\mathbf{B}]^T\{\mathbf{N}\} - \{\beta\} = 0 \tag{1.5}$$

The above equation represents the n deformations of the structure which are expressed in terms of the m displacements or deformations, which leads to $n - m$ constraints on the deformations of the corresponding elements of the structure. These constraints are known as **Compatibility conditions**. They are formally expressed in terms of the **Compatibility Matrix**, denoted by $[\mathbf{C}]$ and $\{\beta\}$ as

$$[\mathbf{C}]\{\beta\} = 0 \tag{1.6}$$

We may denote $\{vect\beta\}$ by the following equation:

$$\{\beta\} = [\mathbf{G}]\{\mathbf{F}\} \tag{1.7}$$

Basic Theory of the Integrated Force Method IV

Combining the equations (1.1) and (1.7), we have,

$$\left[\frac{[\mathbf{B}]}{[\mathbf{C}][\mathbf{G}]}\right]\{\mathbf{F}\} = \left\{\frac{V}{0}\right\} \tag{1.8}$$

The matrices [**B**] and [**G**] are dependent on the equilibrium conditions and the type of material from which the structure is made. The computation of [**C**] matrix, however, is non-trivial and computationally intensive. Moreover, [**C**] is not unique and its structure can vary considerably depending upon the method employed for its generation. Computing, [**C**] for large structures can be quite complicated and time consuming. Thus there is a strong need to automate the task of generating the [**C**] matrix, given the input [**B**] matrix.

Basic Theory of the Integrated Force Method V

For more detailed explanation on the Integrated Force Method, please refer to the following link:

`http://gltrs.grc.nasa.gov/reports/2004/TP-2004-207430.pdf`

Thus, the central aim of this project is auto-generation of compatibility conditions with optimum banded structure, which helps the use of efficient solution techniques for solving IFM governing equation.

Definitions I

Definition (**Integrated Force Methods**)

- The Integrated Force Method (IFM) was proposed by Patnaik (1973) for the analysis of discrete and continuous systems. IFM is a force method of analysis which is independent of redundants and basis determinate structure of the classical force method.

Definition (**Framed/Discrete Structure**)

- Structure supported mainly by a skeleton, or frame, of wood, steel, or reinforced concrete rather than by load-bearing walls. E.g. Truss

Definition (**Continuous Structure**)

- Structure which does not have an underlying skeleton, but is a continuous mass. E.g. Load Bearing Wall

Definitions II

Definition (**Flexibility of a Structure**)

- A flowgram is a data file which consists of readings recorded by the Mass Spectrography Machine. These readings are the intensities of the light emitted by different element atoms in the specimen, when subjected to mass spectrography. Normally a flowgram consists of millions of data values.

Overview of the Problem I

From some background equation of the **Integrated Force Method** for the analysis of non-linear strucutres, we can form our problem as follows, referring to equation (1.8) in matrix form:

$$\begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \dots \\ \theta_n \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & \dots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots & \\ \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_m \end{pmatrix} \tag{1.9}$$

The equation can also be stated in the form

$$\theta = \mathrm{AX} \tag{1.10}$$

Here, $\theta_1, \dots, \theta_n$ - Moments related to the beam
$a_{11}, \dots, a_{nm}$ - Coefficients $x_1, x_2, \dots, x_m$ - One dimensional Row Vectors
The problem here is that the A matrix does not have all the rows required

Overview of the Problem II

in it to make it a square matrix. The A matrix is very important to other parts of the project. We need to compute the inverse of the A matrix in order proceed further in the project, which is possible only when A is a square matrix. Hence, in order to make A a square matrix, we must find new relationships between $x_1, x_2, \ldots, x_n$ so that we can add new rows to the A matrix and form our equation as follows:

$$\begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \\ \theta_n \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{21} & a_{22} & \ldots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{1.11}$$

The Actual Problem

From the above discussion, the problem can be framed as follows:
We need to find the values of scalar coefficients $k_1, k_2, \ldots, k_m$ such that

$$k_1 x_1 + k_2 x_2 + \ldots + k_m x_m = 0 \tag{1.12}$$

In matrix form,

$$\begin{pmatrix} k_1 & k_2 & \ldots & \ldots & k_m \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_m \end{pmatrix} = 0 \tag{1.13}$$

Here $x_1, x_2, \ldots, x_m$ are row vectors of definite length. Basically, we need to find new relationships between the x_is in order to fill the A matrix shown above. This is known as **Generation of Compatibility Conditions**.
We generate these compatibility conditions using matrix manipulations, in order to obtain the desired results.

The Method I

To achieve the task of generating compatibility conditions for the beam, we can adopt the following method: Let $X = x_1, x_2, \ldots, x_m$ Let $\alpha = k_1, k_2, \ldots, k_m$

- First, we split the vector set X into two components:
  1. X_{ind} - The set of all linearly independent vectors $x_{i_{i\epsilon 1,\ldots,m}}$
  2. X_{dep} - The set of all linearly dependent vectors $x_{j_{j\epsilon 1,\ldots,m}}$
- Also the coefficient set α is split up into α_{ind} and α_{dep} in the same manner as the set X. Thus we have the following equation:

$$\alpha_{ind} X_{ind} + \alpha_{dep} X_{dep} = 0 \tag{2.1}$$

- Simplifying futher, we have,

$$\alpha_{ind} X_{ind} = -\alpha_{dep} X_{dep} \tag{2.2}$$

The Method II

- Now we take random values for α_{dep} thus producing a unique value of α_{ind} as follows:

$$\begin{aligned} \alpha_{ind} X_{ind} &= -\alpha_{dep} X_{dep} \\ \therefore \alpha_{ind} (X_{ind}^T X_{ind}) &= -X_{ind}^T \alpha_{dep} X_{dep} \quad (2.3) \\ \therefore \alpha_{ind} &= -(X_{ind}^T X_{ind})^{-1} X_{ind}^T \alpha_{dep} X_{dep} \quad (2.4) \end{aligned}$$

- Thus, we obtain the solution for the coefficient set α as

$$\alpha = [\alpha_{ind} \ \alpha_{dep}]$$

- Once we have the α matrix, we can get the relationship between the $x_{i_{i=1\ldots n}}$. The user can specify the number of solutions/relations wanted, which will be the no. of iterations of the program to be run.

Algorithm for Compatibility Conditions I

Require: B$\leftarrow$ Input X Matrix of size $m * k$, k = no. of elements in each vector x_i

Ensure: B $\neq$ null

1: $X_{ind} \leftarrow [x_1]$
tol $\leftarrow$ (Some very small value, say 0.000001)
$X_{dep} \leftarrow$ null

2: **for** $j = 0$ to m **do**

3: xtest $\leftarrow$ B(:,j) (x_j on the jth iteration)
Since, X_{ind} is the set of lineary independent vectors, we must be able to express xtest in terms of X_{ind}
$\therefore$ xtest = B.X_{ind}

4: B = $(X_{ind}^T X_{ind})^T X_{ind}^T$.xtest

5: Test the result:
result $\leftarrow$ xtest - $(X_{ind} * B)$

6: **if** result$<$tol **then**

Algorithm for Compatibility Conditions II

7: Add the vector to X_{ind}
8: **else**
9: Add the vector to X_{dep}
10: **end if**
11: **end for**

Once we have all the solution coefficients k_i, we can club them together in a single matrix as follows:

$$\text{Solution Matrix } C = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1m} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{s1} & \alpha_{s2} & \dots & \alpha_{sm} \end{pmatrix} \tag{2.5}$$

Here $s =$ no. of solutions wanted by the user.
To test whether this is true or not, we must use another algorithm, which may be stated on the next slide.

Algorithm to test the correctness of the solution obtained

Require: $C \leftarrow$ Solution Matrix

$B \leftarrow$ X matrix

1: B^T = transpose(B)

2: R $\leftarrow C.B^T$

3: The resultant matrix R should be a zero matrix or very close to a zero matrix.

4: Done

Results I

From the above computations, we finally obtain or intend to obtain the solution matrix C which can then be used in the original Solution Matrix A in order to make it a square matrix and have the inverse computation possible on A. i.e. The solution matrix C thus generated using the above algorithm can then be used to add additional rows in the A matrix. The final equation can be described as below:

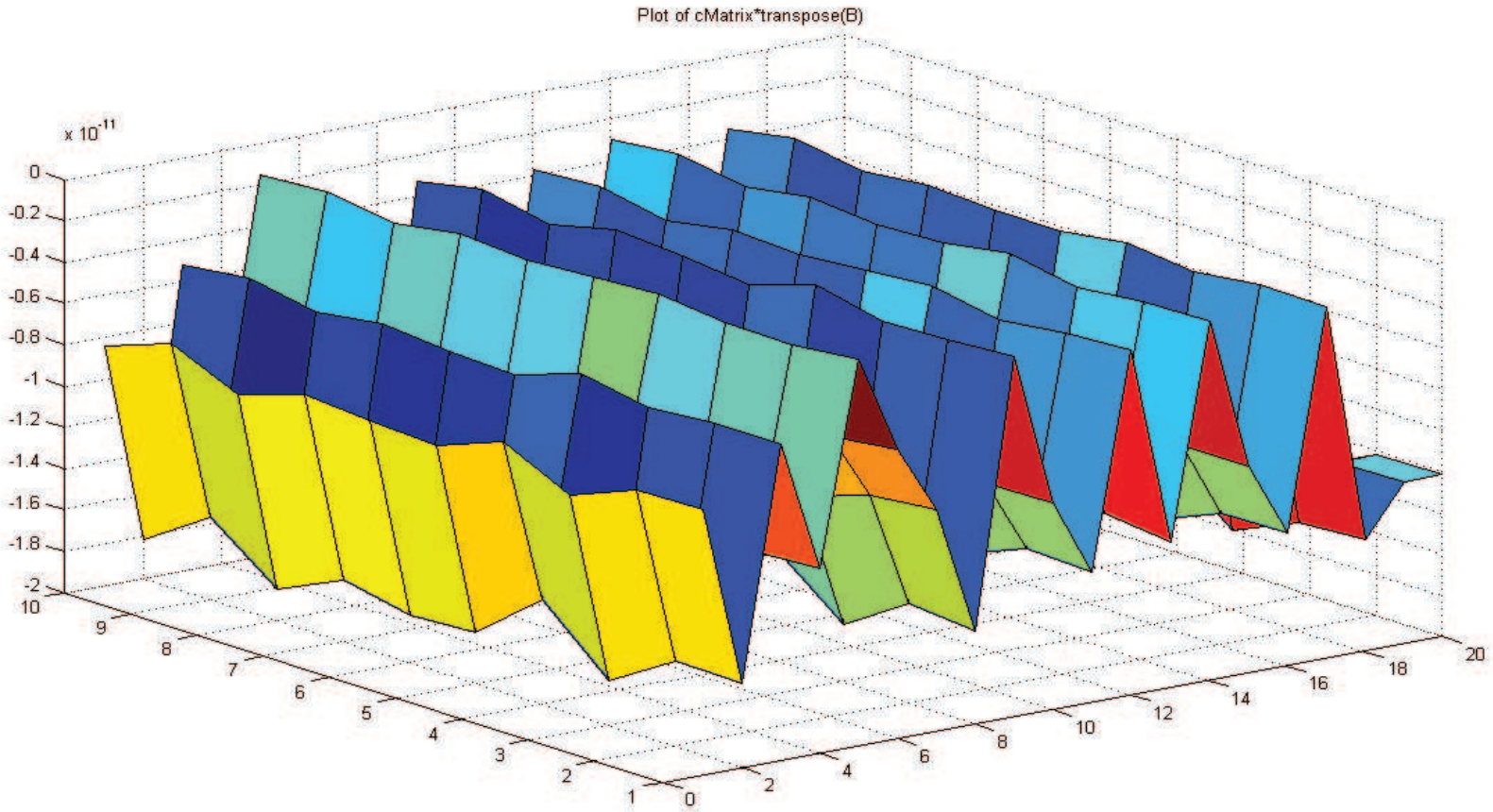
$$\begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \\ \theta_n \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ c_{11} & c_{12} & \dots & c_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ c_{s1} & c_{s2} & \dots & c_{sm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{3.1}$$

Results II

The subsequent pages show results obtained on test runs of the program for different sizes of the input matrix B.

Result : Output I

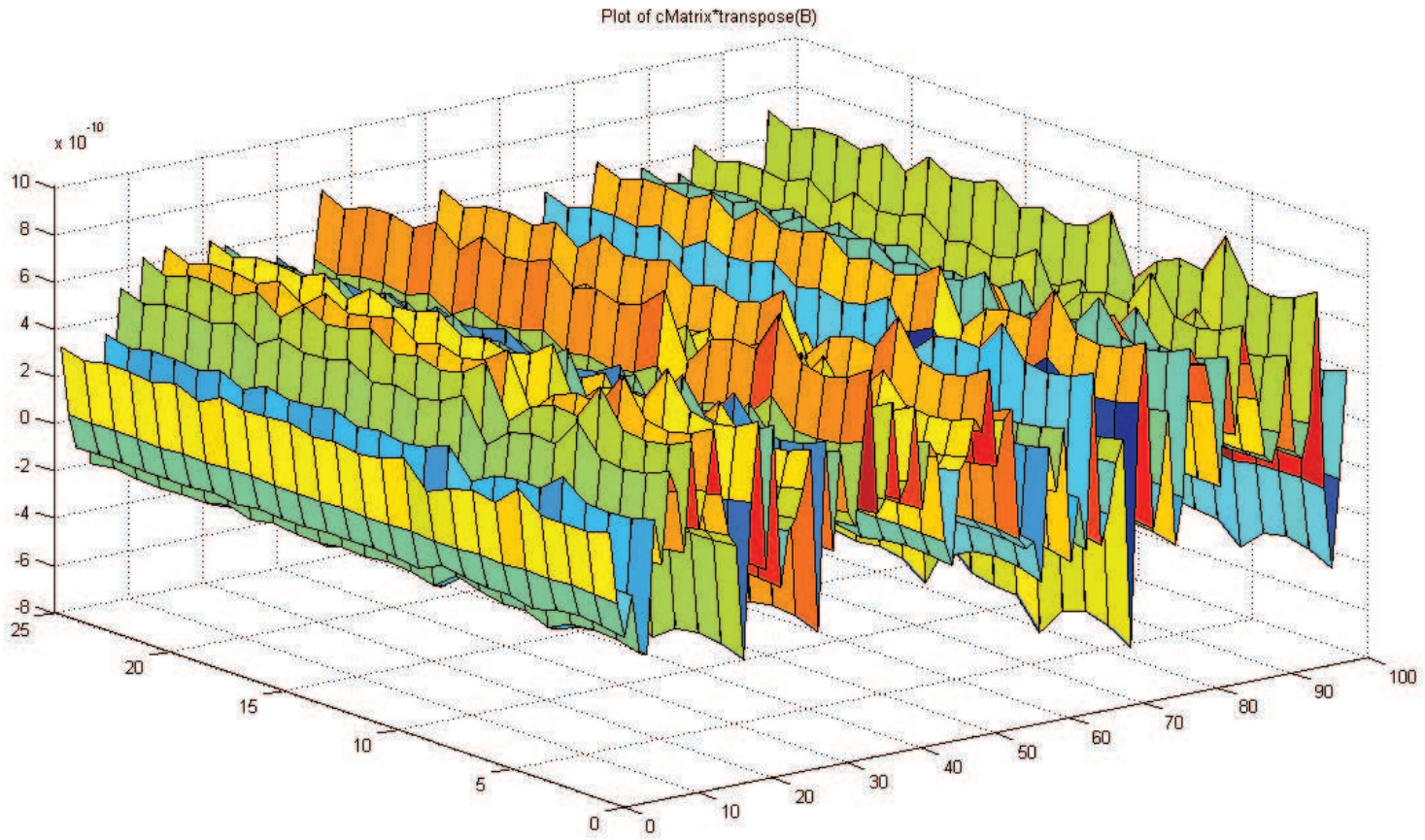
Input Matrix: B = rand(50,90) and no. of solutions = 10



Plot showing the test matrix i.e. C*transpose(B). It is a 10 * 50 Matrix. The x-axis represents the row index, while the y-axis represents both the column index and the corresponding value of a{ij} . As it is evident, the value of the entries of the matrix is almost zero and hence the solution provided by the program is verified

Result : Output II

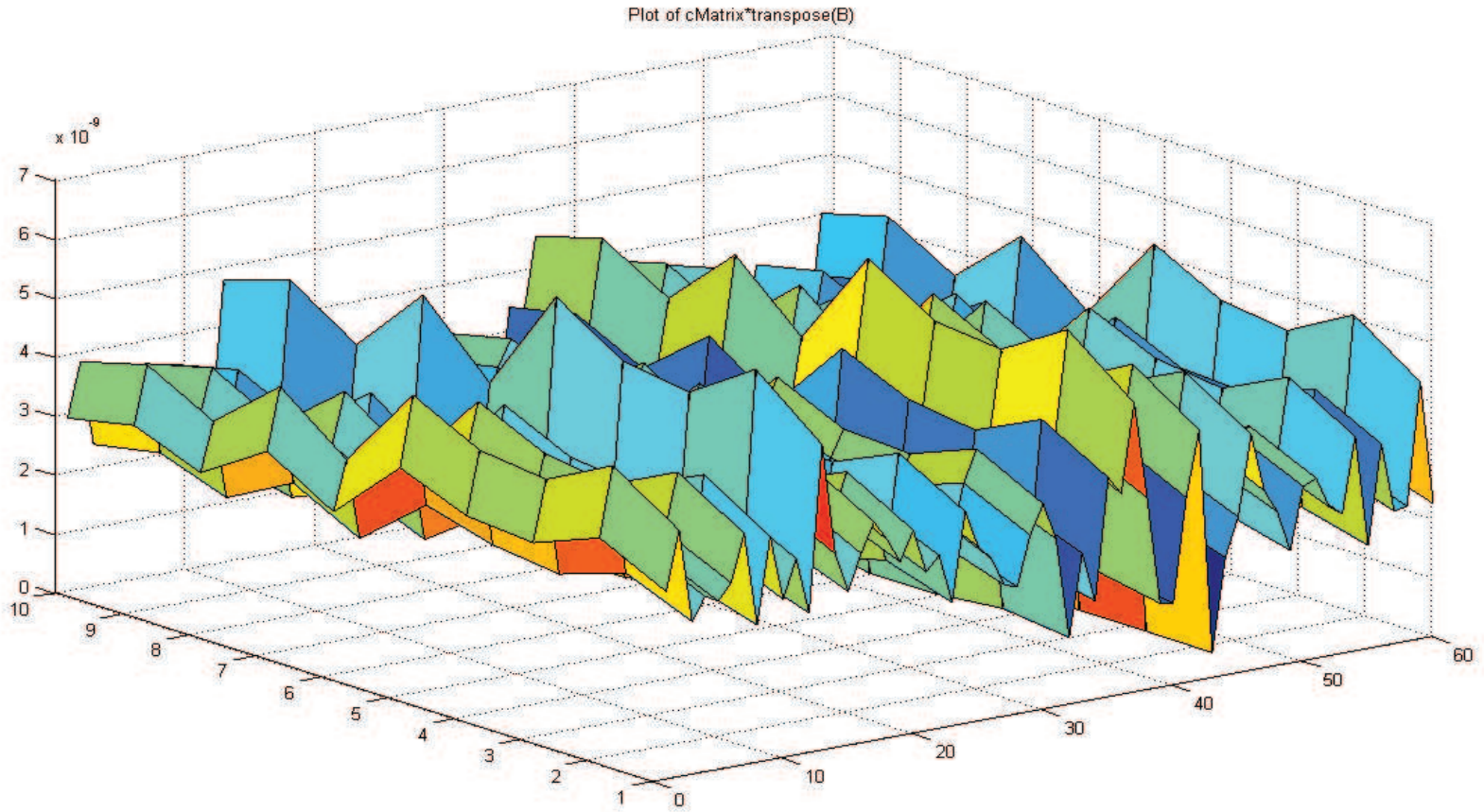
Input Matrix: B = rand(100,150) and no. of solutions = 25



Plot showing the test matrix i.e. C*transpose(B). It is a 25 * 100 Matrix. The x-axis represents the row index, while the y-axis represents both the column index and the corresponding value of a{ij} . As it is evident, the value of the entries of the matrix is almost zero and hence the solution provided by the program is verified

Result : Output III

Input Matrix: B = rand(60,80) and no. of solutions = 10



Plot showing the test matrix i.e. C*transpose(B). It is a 10 * 60 Matrix. The x-axis represents the row index, while the y-axis represents both the column index and the corresponding value of a{ij} . As it is evident, the value of the entries of the matrix is almost zero and hence the solution provided by the program is verified

Discussion

As it is clear from the above results, that the automated computations are quite efficient and much faster as compared to manual computations. Moreover, the results are also highly accurate, as it can be seen from the tests conducted using different source matrices B. The result of $C.B^T$ is almost a zero matrix, with an error of the order of 10^{-7} which is negligible and not even required to be considered.

Thus, using the simple algorithm outlined above, we can easily reduce the computation time required for this process, which ranges from a few weeks to even months of manual labour. The running time of this automated computation is a few seconds. Hence, this saves a lot of time and at the same time it ensures that the results are highly accurate. Hence the entire time can now be devoted to interpreting and evaluating the results rather than just computing them.

Conclusion

Thus, analyzing the results obtained using this method of generating the compatibility matrix [**C**], we can conclude that this method is quite efficient and accurate in doing what it is supposed to do. Moreover, there are drastic savings in terms of the time required to compute the [**C**] matrix, as this method takes only a few seconds to give the accurate results, as compared to the manual method which takes several weeks to complete and there are always doubts about the accuracy of the results. Hence, to conclude, it can be safely stated that this method provides a significant improvement both in terms of computation time and complexity, over the prior methods.