Application of adjustment via Mathematica for deflection of the vertical determination

L. Völgyesi
Department of Geodesy and Surveying
Gy. Popper
Department of Structural Mechanics, Research Group for HAS-BUTE Computational Structural Mechanics
B. Palánz
Department of Photogrammetry and Geoinformatics
Budapest University of Technology and Economics, H-1521 Budapest, Hungary, Müegyetem rkp. 3.

Abstract. In majority of real adjustment problems, the observation equations to be solved in least square sense, have more hundreds or thousands variables, the matrix of the system is sparse and often ill-conditioned. In this case study, the application of integrated, symbolic-numeric system, Mathematica to solve a practical adjustment problem for deflection of the vertical determination is demonstrated. Solving a real world adjustment problem, represented by more thousands variables and equations, with very high sparseness of the system matrix, we illustrate, how easy for practitioners to create and handle sparse matrices economically, compute condition numbers, pseudoinverse and carry out singular value decomposition and solve least square problems with Mathematica within seconds.

Keywords. Adjustment via Mathematica, sparse matrices, singular value decomposition, pseudoinverse solution, deflection of the vertical determination.

Introduction

It is necessary to treat large sparse matrices which may be ill conditioned depending on the geometry of interpolation network in the case of deflection of the vertical’s determination based on torsion balance measurements Völgyesi (1993, 1995, 2001), Tóth, Völgyesi (2002).

However, MATLAB has also advanced numeric methods for sparse matrices, the authors select Mathematica, because of its functional programming ability and perfect interactivity make it ideal for engineers without professional programming knowledge. The interested readers can found a comparison of numeric performances of different mathematical programs for data analysis in Steinhaus (2002) as well as a very good introduction to Mathematica in Ruskeepaa (2004).

1 Mathematical background

Consider the system of linear algebraic equations

\[ Ax = b \]  \hspace{1cm} (1)

where \( A \) is a real \( m \times n \) matrix, \( m > n \) and \( b \) is a real vector of \( m \) elements. If \( m > n \), then the equation (1) is overdetermined and in general its solution does not exist. In similar cases it is advantageous generalize the concept of solution. The vector \( x \) of \( n \) elements, which minimizes the Euclidian norm of the error-vector \( b - Ax \), i.e.

\[ \min_{x} \| b - Ax \|_2 = \| (b - Ax)^T (b - Ax) \|^{1/2} = \min \]  \hspace{1cm} (2)

is called the least-squares solution of equation (1). If the rank of \( A \) is smaller than \( n \), then the minimizing problem (2) has not unique solution. The solution can be made unique if from among all solutions of problem (2) we select that vector \( x^* \) which Euclidian norm is minimal, i.e. \( \| x^* \|_2 = \min \) holds.

This unique least-squares solution of equation (1) can be computed using the formula

\[ x^* = A^+ b \]

where \( A^+ \) is the generalized- (or Moore-Penrose-) inverse of \( A \). In Mathematica \( A^+ \) is implemented using the function Pseudoinverse.

The problem (2) is equivalent to solving so-called normal equations

\[ A^T Ax = A^T b \]

The solution of a least-squares problem directly from the normal equations is rather susceptible to roundoff error.
The most powerful method for computing the generalized- (or Moore-Penrose-) inverse and consequently for least-squares solution of equation (1) too, is based on the singular value decomposition or SVD of matrix $A$.

The SVD methods are based on the following theorem: any real $m \times n$ matrix $A$, $m \geq n$ can be decomposed as

$$A = U\Sigma V^T$$

where $U^T U = V^T V = I$, $\Sigma = \{\sigma_1, \ldots, \sigma_r\}$ is diagonal matrix, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ and $r$ is the rank of $A$.

The $r$ columns of $U$ are the orthonormal eigenvectors of $AA^T$ corresponding to its $r$ largest eigenvalues. The $r$ columns of $V$ are the orthonormal eigenvectors of $A^TA$ corresponding to its $r$ largest eigenvalues. The diagonal elements of $\Sigma$ are square roots of the $r$ largest eigenvalues of $A^TA$ and are called singular values of $A$.

In Mathematica the SVD is implemented using the function `SingularValues`.

Using the SVD factorization, the $A^+$ generalized- (or Moore-Penrose-) inverse of $A$ can be easily computed as follows:

$$A^+ = U \Sigma^+ V^T$$

$$\Sigma^+ = \{1/\sigma_1, \ldots, 1/\sigma_r\}$$

$$A^+ = V \Sigma^+ U^T.$$  

The linear least-squares problem becomes more strongly ill-conditioned as the spectral condition number

$$\text{cond}_s(A) = \sigma_1 / \sigma_r \geq 1$$

increases.

### 2 Data preparation

Computations of deflections of the vertical were performed in a test area in Hungary extending over some 3000 km$^2$ where torsion balance measurements and astrogeodetic data are available.

A simple relationship, based on potential theory, can be written for the changes of $\Delta \xi_{ik}$ and $\Delta \eta_{ik}$ between arbitrary points $i$ and $k$ of the deflection of the vertical components $\xi$ and $\eta$ as well as for curvature values $W_\Lambda = W_{yy} - W_{xx}$ and $2W_{xy}$ measured by a torsion balance (Völgyesi 1993, 1995):

$$\Delta \xi_{ik} \sin \alpha_{ik} + \Delta \eta_{ik} \cos \alpha_{ik} = \xi_k \sin \alpha_{ik} + \eta_k \cos \alpha_{ik} - \xi_i \sin \alpha_{ik} - \eta_i \cos \alpha_{ik} = \frac{s_{ik}}{4g} \left(\left(W_\Lambda - U_\Lambda\right)_i + \left(W_\Lambda - U_\Lambda\right)_k\right) \sin 2\alpha_{ik} \quad (3)$$

$$+ \left(W_{xy} - U_{xy}\right)_i + \left(W_{xy} - U_{xy}\right)_k \cos 2\alpha_{ik}$$

where $s_{ik}$ is the distance between points $i$ and $k$, $g$ is the average value of gravity between them, $U_\Lambda = U_{yy} - U_{xx}$ and $U_{xy}$ are curvature values of the normal (reference) gravity field, whereas $\alpha_{ik}$ is the azimuth between the two points.

The computation being fundamentally integration, practically possible only by approximation; in deriving (3) it had to be assumed that the change of gravity gradients between points $i$ and $k$, measurable by torsion balance, is linear (Völgyesi 1993). This means Eq. (3) results a large sparse linear system.

In order to illustrate the solution methods supported by Mathematica, first we show how to create a sparse matrix object and how to set up the equation system from the input data (Popper, 2003).

The matrix of the equation system can be stored in a file, element by element, in the following way:

<table>
<thead>
<tr>
<th>row number</th>
<th>column number</th>
<th>value</th>
</tr>
</thead>
</table>

Command reading data from a file into a list object is,

```mathematica
MatrixInput = ReadList["D:\Data\Matrix.dat", {Number,Number,Number}];
```

This list object can be converted into a sparse array object as,

```mathematica
Converter[row_,column_,value_] := {row,column} \[Rule] value

A=SparseArray[Map[Converter[#]&,MatrixInput]]
```

where $value$ is the value of the matrix element assigned to the $row$-th row and $column$-th column. Converting `MatrixInput` list into a sparse array, we get

```mathematica
A=SparseArray[Map[Converter[#]&,MatrixInput]]
```

SparseArray object is a list object that can be converted into a sparse matrix object.`
internally using compressed sparse row formats, generalized for tensors of arbitrary rank.

The echo of this command shows us, that \( A \) is a sparse matrix object with 2068 rows and 1462 columns. The number of the nonzero elements are 8218. These can be computed as,

\[
\text{NumberOf NonzeroElements} = \text{Length} [\text{Select} [\text{Flatten} [A], # \neq 0 & ]]
\]

\[
\text{NumberOf AllElements} = \text{Apply} [\text{Times}, \text{Dimensions} [A]]
\]

Consequently the sparseness of the matrix is,

\[
\text{Sparseness} = 1. - \frac{\text{NumberOf NonzeroElements}}{\text{NumberOf AllElements}}
\]

which is: 0.997282.

The right hand side vector of the system can be read similarly, from a different file,

\[
\text{RightHandSideVector} = \text{ReadList} ["D:\Data\Rside.dat", \{\text{Number}\}] / \text{Flatten};
\]

which is a list of 2068 elements.

### 3 Condition Number

Looking for a solution in sense of least squares, the pseudo matrix is,

\[
\text{PseudoMatrix} = \text{Transpose} [A].A;
\]

its condition number based on \( p \)-norm, \( p = 2 \), is

\[
\text{MatrixConditionNumber} [\text{PseudoMatrix}, 2]
\]

which is 2107.78. This means that the PseudoMatrix matrix is not an ill conditioned matrix, therefore pseudoinverse solution can be employed.

### 4 Displaying the structure

Matrix structure can be graphically displayed via the \text{MatrixPlot} functionality. The array is shown as a grid of black and white cells, by default representing zero-valued cells as white, and non-zero values as black. \text{MatrixPlot} accept the usual range of options for a graphics function. In addition, it takes the MaxMatrixSize option, specifying a maximum display size to use for the array. By default, this is set to 512. If the matrix dimensions are larger than this size, the matrix is downsampled to this size or less so that the output graphic size can be controlled. In this case, a darkened cell indicates that at least one of the covered cells has a non-central value. The MaxMatrixSize may be set to \text{Infinity} to prevent downsampling before display,

\[
<< \text{LinearAlgebra}':\text{MatrixManipulation'}
\]

\[
\text{MatrixPlot} [A, \text{MatrixSize} \rightarrow \text{Infinity}];
\]

\[\text{Fig. 1 The structure of the sparse matrix}\]

### 5 LinearSolve solution

First we try the \text{LinearSolve} function works on both numerical and symbolic matrices, as well as \text{SparseArray} objects. For sparse arrays, \text{LinearSolve} uses UMFPACK multifrontal direct solver methods and with Method->"Krylov" uses Krylov iterative methods preconditioned by an incomplete LU factorization only for numeric matrices.

\[
\text{Timing} \left[ \text{LinearSolve} [\text{PseudoMatrix}, \text{Transpose} [A].\text{RightHandSideVektor}] \right];
\]

\[\{0. \text{ Second}, \text{Null}\}\]

or with the Krylov method

\[
\text{Timing} \left[ \text{LinearSolve} [\text{PseudoMatrix}, \text{Transpose} [A].\text{RightHandSideVektor}, \text{Method} \rightarrow \text{Krylov}] \right];
\]

\[\{0.031 \text{ Second}, \text{Null}\}\]
The solution vector

\[ x = \text{LinearSolve}[\text{PseudoMatrix}, \text{Transpose}[A].\text{RightHandSideVektor}]; \]

ListPlot[x, PlotJoined -> True, Frame -> True, PlotRange -> {-3, +3}, FrameLabel -> {"Number of variables", "Value of variable"}];

4 SVD Solution

The \text{SingularValueDecomposition} function gives the singular value decomposition for a numerical matrix \( A \). The result is a list of matrices \( \{u, \sigma, v\} \), where \( \sigma \) is a diagonal matrix, and \( A \) can be written as \( u \cdot \sigma \cdot \text{ConjugateTranspose}[v] \).

\text{SingularValueDecomposition} uses the QR decomposition algorithm of Francis with Given's rotations.

\text{Timing}[\text{SingularValues}[A];]
\{42.781 \text{ Second}, \text{Null}\}

which is greater with two magnitude than running time of \text{LinearSolve}.

\( \{u, \sigma, v\} = \text{SingularValues}[A]; \)

The diagonal matrix is

\( \Xi_r = \text{DiagonalMatrix}[\sigma]; \)

The \( p = 2 \) norm of a matrix is the largest principal axis of the ellipsoid, equal to the largest singular value of the matrix.

\text{Norm}[A] == \text{Max}[\sigma]
True

The spectral condition number gives a better estimation for the ill-conditioning, than condition number based on \( p = 2 \) norm

\( \sigma[[1]]/\text{Last}[\sigma] \)
45.9466

The matrices of \( u \) and \( v \) are

\( \{U_r, V_r\} = \text{Map}[\text{Transpose}, \{u, v\}]; \)

The matrix \( A \) can be expressed with these matrices and with the diagonal matrix, consequently

\text{Norm}[A- U_r \cdot \Xi_r \cdot \text{Transpose}[V_r]]
1.98418 \times 10^{-14}

The inverse matrix is

\( A^* = V_r \cdot \text{Inverse}[\Xi] \cdot \text{Transpose}[U_r]; \)

To check round of errors we can express \( A \) with the inverse, too

\text{Norm}[A-A \cdot A^* \cdot A]
1.80475 \times 10^{-14}
The solution vector is

\[ xx = A^\dagger \cdot \text{RightHandSideVektor}; \]

The difference between the first and second solution method is

\[ \text{Norm}[x - xx] \]

\[ = 9.7108 \times 10^{-11} \]

### 7 Pseudoinverse solution

The third solution method computes the pseudoinverse of \( A \) directly, indeed

\[ \text{Norm}[\text{PseudoInverse}[A] - A^\dagger] \]

\[ = 2.11709 \times 10^{-14} \]

For numerical matrices, \text{PseudoInverse} is based on \text{SingularValueDecomposition}, therefore the time of this computation is very close to that of the SVD method,

\[ \text{Timing}[(\text{PseudoInverse}[A] \cdot \text{RightHandSideVektor});] \]

\[ \{43.563 \text{ Second, Null}\} \]

which is greater again with two magnitude than running time of \text{LinearSolve}. The solution vector is

\[ xxx = \text{PseudoInverse}[A] \cdot \text{RightHandSideVektor}; \]

The total residuum of the third solution is

\[ \text{Norm}[A \cdot xxx - \text{RightHandSideVektor}]; \]

\[ = 6.68605 \]

Comparing the first and third solutions, the norm of the difference of the solution vectors is

\[ \text{Norm}[x - xxx] \]

\[ = 9.71054 \times 10^{-11} \]

However, the execution time of SVD type methods are significantly greater than that of the \text{LinearSolve}, one can use them in case of ill conditioned problem, too.

### 8 Complexity Study

It is important to know, how the computation time increases with the increase of \( n \). According to our computation carried out on PC Compaq Evo P4 2.8 MHz with \text{Mathematica} Version 5, the Table 1 shows the results of two characteristic runs where the computation time is in seconds.

<table>
<thead>
<tr>
<th>Matrix A</th>
<th>Sparse-ness</th>
<th>Linear Solve</th>
<th>Pseudo Inverse</th>
<th>SVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>559×400</td>
<td>0.990</td>
<td>0.031</td>
<td>1.188</td>
<td>1.203</td>
</tr>
<tr>
<td>2068×1462</td>
<td>0.997</td>
<td>0.130</td>
<td>43.563</td>
<td>42.781</td>
</tr>
</tbody>
</table>

### 9 Interpolated deflections of vertical

Interpolated N-S (\( \xi \)) and E-W (\( \eta \)) components of deflections of the vertical that resulted from the computation visualized on isoline map in Figure 6 and 7. Isoline interval is 0.2". The interpolation network has 738 torsion balance stations (marking by dots in figures) and 731 of these are points with unknown deflection of the vertical. Since there are two unknown components of deflection of the vertical at each point there are 1462 unknowns for which 2068 equations can be written. From these 738 torsion balance stations there were 11 astrogeodetic and astrogravimetric points where \( \xi, \eta \) values were known referring to the \text{GRS80} system. 7 astrogeodetic points were used as initial (fixed) points of interpolation and 4 points were used for checking of computations.

Standard deviations \( m_\xi = \pm 0.60'' \) and \( m_\eta = \pm 0.65'' \), computed at checkpoints confirm the fact that \( \xi, \eta \) values of acceptable accuracy can be computed from torsion balance measurements and \text{Mathematica} can be efficiently applied for solving this adjustment problem.

![Fig. 6 Computed N-S (\( \xi \)) component of deflections of the vertical](image-url)
Fig. 7 Computed E-W ($\eta$) component of deflections of the vertical

10 Conclusions

LinearSolve based on UMFPACK multifrontal direct solver methods and with Method->"Krylov" uses Krylov iterative methods preconditioned by an incomplete LU factorization as well as the PseudoInverse uses SVD method, give the same result, however the running time of the latest is greater with about two magnitudes. In case of bad conditioned pseudo matrix, PseudoInverse function is recommended. Using a standard PC like we did, we get solution under realistic time (approx. 40 seconds) in case of a not bad conditioned system matrix, for $n = 1462$ variables. It goes without saying that these limits could be doubled with PC employing 64 bits processors.

Acknowledgements

We should thank for the funding of the above investigations to the National Scientific Research Fund (OTKA T-037929 and T-37880), and for the assistance provided by the Physical Geodesy and Geodynamic Research Group of the Hungarian Academy of Sciences.

References


* * *


Dr. Lajos VÖLGYESI, Department of Geodesy and Surveying, Budapest University of Technology and Economics, H-1521 Budapest, Hungary, Műegyetem rkp. 3.
Web: http://sci.fgt.bme.hu/volgyesi  E-mail: volgyesi@eik.bme.hu