The Sherman–Morrison Formula

Susanne Brenner and Li-Yeng Sung
(modified by Douglas B. Meade)
Department of Mathematics

Overview

To solve $\mathbf{Bx} = \mathbf{b}$ when $\mathbf{B}$ is an $n \times n$ matrix using Gaussian elimination with partial pivoting requires $O(n^3)$ operations. If $\mathbf{B}$ has a special form, e.g., triangular or tridiagonal, the computational complexity can be reduced to $O(n^2)$ or $O(n)$. The Sherman–Morrison formula provides an effcient algorithm for solving $\mathbf{Ax} = \mathbf{b}$ when $\mathbf{A}$ is almost one of the special forms for which a more efficient solution algorithm exists.

The primary goal of this lab is to implement the Sherman–Morrison formula. This implementation does not require the creation of an M-file, but does utilize the `forward` and `trisolve` commands implemented in earlier labs.

Part I

The Sherman–Morrison formula provides an explicit formula for the inverse of a matrix $\mathbf{A} = \mathbf{B} - \mathbf{uv}^t$ where $\mathbf{B}$ is a nonsingular $n \times n$ matrix and $\mathbf{u}$ and $\mathbf{v}$ are column n-vectors. First, $\mathbf{A}$ is nonsingular if and only if $1 - \mathbf{v}^t(\mathbf{B}^{-1}\mathbf{u}) \neq 0$ and, in this case, the Sherman–Morrison formula is

$$\mathbf{A}^{-1} = \left(\mathbf{I} + \frac{1}{1 - \mathbf{v}^t(\mathbf{B}^{-1}\mathbf{u})}(\mathbf{B}^{-1}u)\mathbf{v}^t\right)\mathbf{B}^{-1}.$$

<u>Sherman-Morrison Algorithm</u>
The following algorithm for solving $\mathbf{Ax} = \mathbf{0}$ is based on the Sherman–Morrison formula:

STEP 1. Solve $\mathbf{Bz} = \mathbf{u}$.

STEP 2. Compute $\gamma = 1 - \mathbf{v}^t(\mathbf{B}^{-1}\mathbf{u}) = 1 - \mathbf{v}^t\mathbf{z}$.

- If $\gamma = 0$ then $\mathbf{A}$ is singular and $\mathbf{A}^{-1}$ does not exist.
- If $\gamma \neq 0$ then $\mathbf{A}$ is nonsingular; continue with Step 2.

STEP 3. Solve $\mathbf{By} = \mathbf{b}$.

STEP 4. Compute $\beta = \frac{\mathbf{v}^t\mathbf{y}}{\gamma}$.

STEP 5. The solution to $\mathbf{Ax} = \mathbf{B}$ is $\mathbf{x} = \mathbf{y} + \beta\mathbf{z}$.

Observe that this algorithm requires the solution of two systems involving $\mathbf{B}$ (Steps 1 and 3), two inner products (Steps 2 and 4), a scalar-scalar division (Step 2), and a scalar-vector multiplication (Step 5). The number of floating-point operations required to complete this algorithm is of the same order as finding the solution to a linear system with coefficient matrix $\mathbf{B}$.

<u>An Example</u>
Consider the system

$$\begin{bmatrix} 1 & 0 & 0 & -1 \\ -1 & 1 & 0 & 2 \\ 2 & 0 & 1 & 1 \\ 0 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

where

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} 1 \\ -2 \\ -1 \\ 0 \end{bmatrix}, \quad \text{and} \quad \mathbf{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Notice that $\mathbf{B}$ is a lower triangular matrix.

Create **A**, **B**, **u**, **v**, and **b** in MATLAB. Then execute the following commands:

≫ **A - (B-u*v')**	**% check that** $\mathbf{A} = \mathbf{B} - \mathbf{u}\mathbf{v}^t$
≫ **z = forward(B, u)**	**% Step 1: B lower-triangular so** $z = \mathbf{B}^{-1}\mathbf{u}$
≫ **gamma = 1 - v'*z**	**% Step 2: if** $\gamma = 0$**, then A is singular − STOP!**
≫ **y = forward(B, b)**	**% Step 3: solve** $\mathbf{B}\mathbf{y} = \mathbf{b}$ **(B lower triangular)**
≫ **beta = (v'*y)/gamma**	**% Step 4:**
≫ **x = y + beta*z**	**% Step 5: solution to** $\mathbf{A}\mathbf{x} = \mathbf{b}$
≫ **A*x-b**	**% check accuracy of solution**

Clear all variables before you begin to work on Part II.