Linear Matrix Equations – Part 1

Mark A. Austin

University of Maryland

austin@umd.edu
ENCE 201, Fall Semester 2023

November 3, 2023

Overview

Linear

Matrix Equations

Linear Matrix Equations

Definition. A system of m linear equations with n unknowns may be written

$$
\begin{array}{ccccccccc}
a_{11}x_1 & + a_{12}x_2 & + & a_{13}x_3 + & \cdots & + & a_{1n}x_n & = & b_1 \\
a_{21}x_1 & + a_{22}x_2 & + & a_{23}x_3 + & \cdots & + & a_{2n}x_n & = & b_2 \\
\vdots & \vdots & & \vdots & & & \ldots & & \vdots \\
a_{m1}x_1 & + a_{m2}x_2 & + & a_{m3}x_3 + & \cdots & + & a_{mn}x_n & = & b_m
\end{array} \tag{1}
$$

Points to note:

- The constants $a_{11}, a_{21}, a_{31}, \cdots a_{mn}$ and $b_1, b_2, \cdots b_m$ are called the equation coefficients.
- The variables $x_1, x_2 \cdots x_n$ are the unknowns in the system of equations.

Linear Matrix Equations

Matrix Form. The matrix counterpart of 1 is $[A] \cdot [X] = [B]$, where

$$[A] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \vdots \\ \vdots & & & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \quad (2)$$

Points to note:

- Matrices A and X have dimensions $(m \times n)$ and $(n \times 1)$, respectively.
- Column vector B has dimensions $(m \times 1)$.

Augmented Matrix Form

Augmented Matrix Form. An augmented matrix for a system of equations is matrix A juxtiposed with matrix B.

Example. The augmented matrix form form of equation 2 is:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & & \vdots & b_2 \\ \vdots & & & \vdots & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} & b_m \end{array}\right] \tag{3}$$

The augmented matrix dimensions are $(m \times (n+1))$.

Definition of Linear

Mathematical Definition. Let k be a non-zero constant. A function y = f(x) is said to be linear if it satisfies two properties:

- $y = f(kx_1)$ is equal to $y = kf(x_1)$.
- $f(x_1 + x_2) = f(x_1) + f(x_2)$.

For constants k and m these equations can be combined:

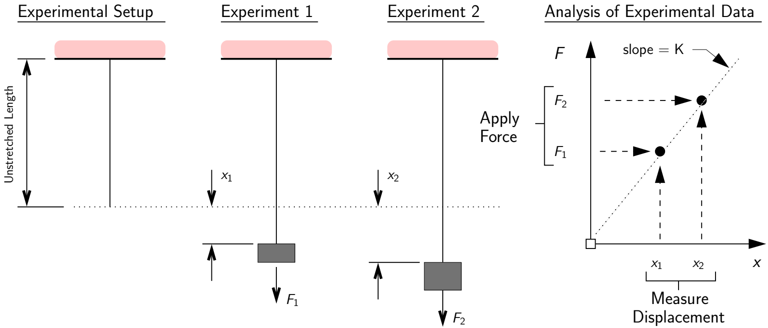
$$kf(x_1) + mf(x_2) \to f(kx_1 + mx_2). \tag{4}$$

Economic Benefit. Often evaluation of $y = f(x)$ has a cost.

Linearity allows us to compute $y_1 = f(x_1)$ and $y_2 = f(x_2)$ and then predict the system response for $kx_1 + mx_2$ via linear combination of solutions. This is free!

Definition of Linear

Example 1. Consider an experiment to determine the extension of an elastic chord as a function of applied force.



Linearity allows us to predict solutions:

$$Kx_1 = F_1, Kx_2 = F_2, \rightarrow K(mx_1 + nx_2) = mF_1 + nF_2. \qquad (5)$$

Definition of Linear

Example 2. Analysis of Linear Structural Systems (ENCE 353):

Let matrix equations $AX = B$ represent behavior of a structural system:

- Matrix A will capture the geometry, material properties, etc.
- Matrix B represents externally applied loads (e.g., dead/live gravity loads).
- Column vector X represents nodal displacements.

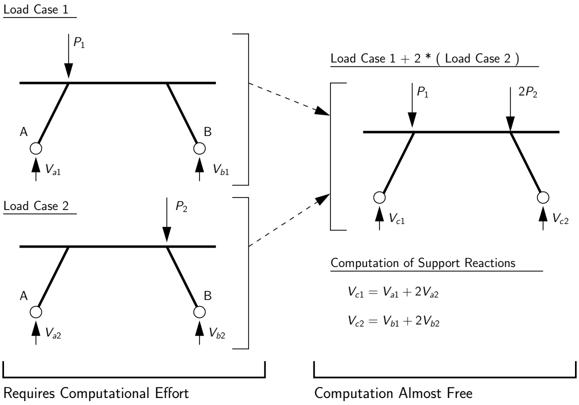
Solving $AX = B$ requires computational work $O(n^3)$.

However, if matrix system is linear, then:

$$AX_1 = B_1, AX_2 = B_2 \rightarrow A(mX_1 + kX_2) = mB_1 + kB_2. \quad (6)$$

Definition of Linear

We can simply add the results of multiple load cases:



Works for support reactions, bending moments, displacements, etc.

Solutions in Two

and

Three Dimensions

Equations in Two Dimensions

Let $m = n = 2$.

The pair of equations:

$$a_{11}\,x_1 + a_{12}\,x_2 = b_1 \tag{7}$$
$$a_{21}\,x_1 + a_{22}\,x_2 = b_2 \tag{8}$$

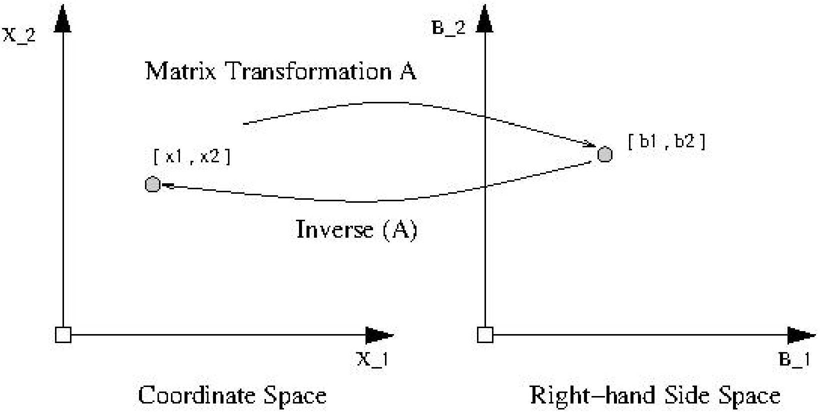
can be interpreted as a pair of straight lines in the (x_1, x_2) plane.

The equations in matrix form are:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{9}$$

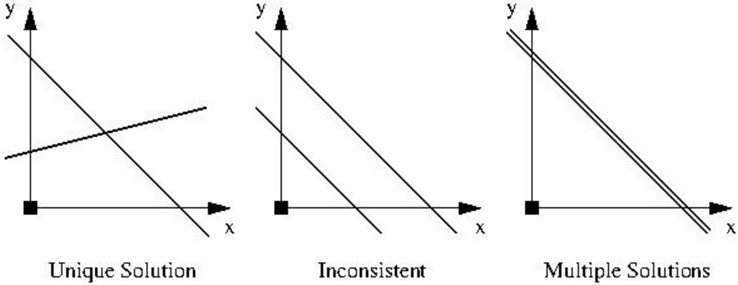
Equations in Two Dimensions

Matrix Tranformation: $[A]\,[X] \rightarrow [B]$.



Equations in Two Dimensions

Three Types of Solutions:



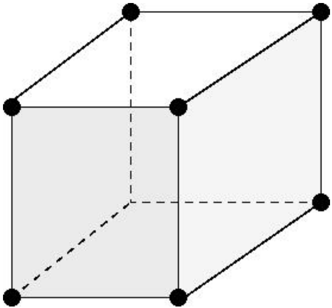
- Unique solution when two lines meet at a point.
- No solutions when two lines are parallel but not overlapping.
- Multiple solutions when two lines are parallel and overlap.

Equations in Three Dimensions

Also Three Types of Solutions:

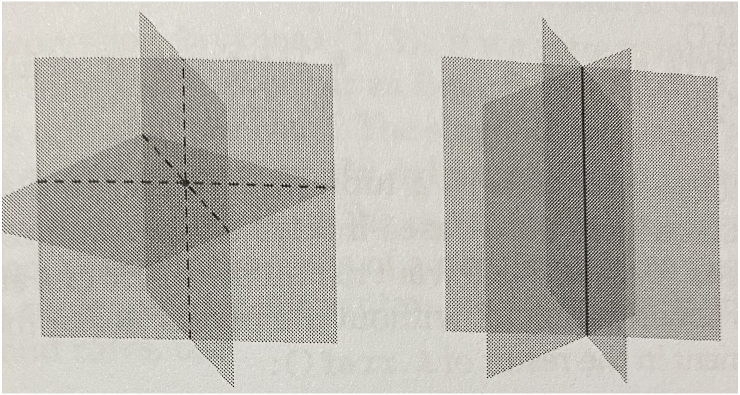
Each equation corresponds to a plane in three dimensions (e.g., think walls, floor and ceiling in a room).

- Unique solution when three planes intersect at a corner point.
- Multiple solutions where three planes overlap or meet along a common line.
- No solutions when three planes are parallel, but distinct, or pairs of planes that intersect along a line (or lines).



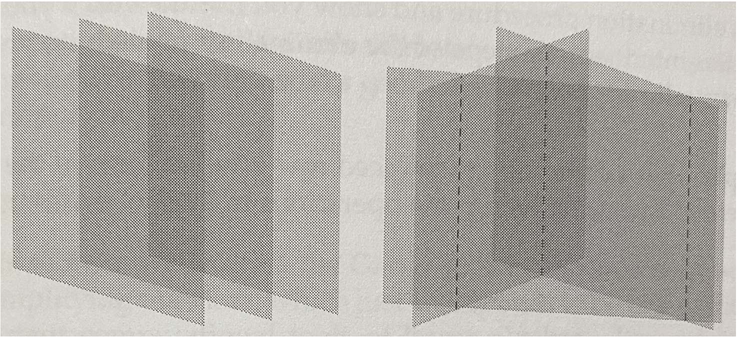
Equations in Three Dimensions

One Solution/Infinite Solutions:



Equations in Three Dimensions

No Solutions:



Analysis of Solutions to Matrix Equations

Key Observations

- For two- and three-dimensions, graphical methods and intuition work well.
- For problems beyond three dimensions, much more difficult to understand the nature of solutions to linear matrix equations.
- We need to rely on mathematical analysis instead.

Basic Questions

- How many solutions will a set of equations will have?
- How to determine when no solutions exist?
- If there is more than one solution, how many solutions exist?

Fortunately, hand calculations on very small systems can provide hints on a pathway forward.

Matrix Determinant

Preample to Matrix Determinant

Strategy. Understand this problem by premultiplying the equations by constants in such a way that when they are combined variables will be eliminated.

Hand Calculation 1: Multiply equation 7 by a_{21} and equation 8 by a_{11}. This gives:

$$\begin{aligned} a_{21} \cdot a_{11} \cdot x_1 &+ a_{21} \cdot a_{12} \cdot x_2 = a_{21} \cdot b_1 \quad (10) \\ a_{11} \cdot a_{21} \cdot x_1 &+ a_{11} \cdot a_{22} \cdot x_2 = a_{11} \cdot b_2 \quad (11) \end{aligned}$$

Next, subtract equation 10 from equation 11 and rearrange:

Preample to Matrix Determinant

$$x_2 = \left[\frac{a_{11} \cdot b_2 - a_{21} \cdot b_1}{a_{11} \cdot a_{22} - a_{12} \cdot a_{21}} \right]. \quad (12)$$

Finally, get x_1 by back-substituting x_2 into either equation 7 or 8.

Turns out there is more than one way to compute a solution ...

Hand Calculation 2: Multiply equation 7 by a_{22} and equation 8 by a_{12}, then subtract and rearrange:

$$x_1 = \left[\frac{a_{22} \cdot b_1 - a_{12} \cdot b_2}{a_{11} \cdot a_{22} - a_{12} \cdot a_{21}} \right] \quad (13)$$

Compute x_2 by back-substituting x_1 into either equation 7 or 8.

Preample to Matrix Determinant

Key Point. The denominators of equations 12 and 13 are the same.

They correspond to the **determinant** of a (2×2) matrix, namely:

$$det(A) = det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}. \qquad (14)$$

Note. The family of equations will have a unique solution when $\det(A) \neq 0$.

Preample to Matrix Determinant

Equations in Three Dimensions. (i.e., $m = n = 3$),

$$det(A) = \ det \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = a_{11}M_{11} - a_{12}M_{12} + a_{13}M_{13}. \tag{15}$$

where,

$$M_{11} = \begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix}, M_{12} = \begin{bmatrix} a_{21} & a_{23} \\ a_{31} & a_{23} \end{bmatrix}, M_{13} = \begin{bmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}. \tag{16}$$

Again, a unique solution exists when det(A) $\neq 0$. det(A) will be zero when two or more planes are parallel.

Matrix Determinant

General Formula. Let A be a $(n \times n)$ matrix.

For each a_{ij} there is a sub-matrix A'_{ij} obtained by deleting the i-th row and j-th column of A.

Let $M_{ij} = \det\,(A'_{ij})$.

i-th row expansion

$$\det(A) = \sum_{j=1}^{n} (-1)^{(i+j)} M_{ij}.$$

j-th row expansion

$$\det(A) = \sum_{i=1}^{n} (-1)^{(i+j)} M_{ij}.$$

$$\begin{array}{c|ccccc} (-1)^{(i+j)} & & & & & \\ \hline & + & - & + & - & + \\ & - & + & - & + & - \\ & + & - & + & - & + \\ & - & + & - & + & - \end{array}$$

Matrix Determinant

Example 1. The most straight forward way of computing the determinant of:

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 3 & -1 & 1 \\ 4 & 6 & -2 \end{bmatrix} \tag{17}$$

is to expand terms about the row or column having the most zero elements – in this case, the first row. This gives:

$$\det(\mathrm{A}) = 2\det \begin{bmatrix} -1 & 1 \\ 6 & -2 \end{bmatrix} = 2(2-6) = -8. \tag{18}$$

Because det(A) evaluates to a non-zero number, we expect that the inverse of A will exist, and as such, the rank(A) = 3.

Matrix Determinant

Example 2. Compute the determinant of:

$$A = \begin{bmatrix} 2 & -1 & 3 & 0 \\ 4 & -2 & 7 & 0 \\ -3 & -4 & 1 & 5 \\ 6 & -6 & 8 & 0 \end{bmatrix} \tag{19}$$

To minimize computation we expand terms about the row or column having the most zero elements – in this case, the third column. This gives:

$$\det(\mathrm{A}) = -5\det \begin{bmatrix} 2 & -1 & 3 \\ 4 & -2 & 7 \\ 6 & -6 & 8 \end{bmatrix} = -5M_{34}. \tag{20}$$

Matrix Determinant

Expanding the second determinant about the first row gives:

$$\begin{aligned} M_{34} &= 2\det\begin{bmatrix} -2 & 7 \\ -6 & 8 \end{bmatrix} + \det\begin{bmatrix} 4 & 7 \\ 6 & 8 \end{bmatrix} + 3\det\begin{bmatrix} 4 & -2 \\ 6 & -6 \end{bmatrix}, \\ &= 2(-16+42) + (32-42) + 3(-24+12) \\ &= 6. \end{aligned} \tag{21}$$

Equation 20 evaluates to -5(6) = -30.

Matrix Determinant

Matrix Determinant Properties:

- $\det(AB) = \det(A)\det(B) = \det(BA)$.
- If $\det(A) \neq 0$, then A is invertable (non-singular).
- $\det(A^T) = \det(A)$.
- det(A) is multiplied by -1 when two rows of A are swapped.
- If two rows of A are identical, then $\det(A) = 0$.

Echelon Form

Echelon Form

Definition

- The first no-zero entry of a row (or column) is called the leading entry.

Definition of Echelon Form

A matrix is in echelon form (i.e., upper triangular form) if:

- All non-zero rows are above any zero row (i.e., a row with all zeros).
- For any two rows, the column containing the leading entry of the upper row is on the left of the column containing the leading entry of the lower row.

Matrices in echelon form display an upper triangular pattern.

Echelon Form

Example 1. Two matrices in echelon form:

$$\begin{bmatrix} 1 & 4 & 0 & 6 & 10 & 6 \\ 0 & 2 & 1 & 0 & 2 & 1 \\ 0 & 0 & 3 & 2 & 1 & 2 \end{bmatrix}, \quad \begin{bmatrix} 1 & 4 & 0 & 6 & 10 & 6 \\ 0 & 2 & 1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{22}$$

Example 2. These matrices are not in echelon form:

$$\begin{bmatrix} 1 & 4 & 0 & 6 & 10 & 6 \\ 0 & 2 & 1 & 0 & 2 & 1 \\ 0 & 3 & 1 & 2 & 1 & 2 \end{bmatrix}, \quad \begin{bmatrix} 0 & 2 & 1 & 0 & 2 & 1 \\ 1 & 4 & 0 & 6 & 10 & 6 \\ 0 & 0 & 1 & 2 & 1 & 2 \end{bmatrix}. \tag{23}$$

Reduced Echelon Form

Definition of Reduced Echelon Form

A matrix is in reduced echelon form if in addition to the criteria stated above:

- All leading entries are 1, and they are the only non-zero entries in each pivot (i.e., leading entry) column.

Any matrix can be reduced by a sequence of elementary row operations to a unique reduced Echelon form.

Example 1. Matrices in reduced Echelon form:

$$\begin{bmatrix} 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 2 & 0 & 10 & 6 \\ 0 & 1 & 1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{24}$$

Elementary

Row Operations

Elementary Row Operations

Purpose. An elementary row operation transforms the structure of matrix equations $[A]\,[X] = [B]$ without affecting the underlying solution $[X]$.

Three Types of Elementary Row Operation

- Swap any two rows.
- Multiply any row by a non-zero number.
- Add to one equation a non-zero multiple of another equation.

Are they Useful?

- Yes! Elementary row operations are used in Gaussian Elimination to reduce a matrix $[A]$ to row echelon form (much easier to work with).

Elementary Row Operations

Example 1. Swap rows 1 and 3:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \xrightarrow{r_1 \leftrightarrow r_3} \begin{bmatrix} g & h & i \\ d & e & f \\ a & b & c \end{bmatrix}. \tag{25}$$

Example 2. Replace row 2 by itself minus 2 times row 1:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \xrightarrow{r_2 \rightarrow r_2 - 2r_1} \begin{bmatrix} a & b & c \\ d-2a & e-2b & f-2c \\ g & h & i \end{bmatrix}. \tag{26}$$

Elementary Row Operations

Row Operations Modeled as Matrix Transformations. Each of these operations can be viewed in terms of an elementary matrix transformation $[E]$, e.g.,

$$A_0 \xrightarrow{\text{row operation}} A_1 \Longleftrightarrow E A_0 \rightarrow A_1. \tag{27}$$

We can design a sequence of transformation matrices E_1, E_2 ... E_n, i.e.,

$$[E_n] \cdots [E_2] [E_1] [A] [X] = [E_n] \cdots [E_2] [E_1] [B], \tag{28}$$

to simplify (upper triangular form) the matrix structure of the left-hand side.

Elementary Row Operations

Example 1. This transformation that swaps rows 1 and 3.

$$[E]\,[A] = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \rightarrow \begin{bmatrix} g & h & i \\ d & e & f \\ a & b & c \end{bmatrix}. \tag{29}$$

Example 2. The matrix transformation:

$$\begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \rightarrow \begin{bmatrix} a+2d & b+2e & c+2f \\ d & e & f \\ g & h & i \end{bmatrix}. \tag{30}$$

replaces row 1 by itself + two times row 2.

Elementary Row Operations

Example 3. Starting from the augmented matrix $[A|I]$, we can design sequences of elementary row operations to compute $\left[I|A^{-1}\right]$, i.e.,

$$[A|I] \xrightarrow{\text{row operations}} \left[I|A^{-1}\right]. \tag{31}$$

Here is a simple example:

$$\left[\begin{array}{rrr|rrr} -1 & 1 & 2 & 1 & 0 & 0 \\ 3 & -1 & 1 & 0 & 1 & 0 \\ -1 & 3 & 4 & 0 & 0 & 1 \end{array}\right] \xrightarrow{\text{row ops}} \left[\begin{array}{rrr|rrr} 1 & 0 & 0 & -0.7 & 0.2 & 0.3 \\ 0 & 1 & 0 & -1.3 & -0.2 & 0.7 \\ 0 & 0 & 1 & 0.8 & 0.2 & -0.2 \end{array}\right]. \tag{32}$$

Elementary Row Operations

Example 4. The computational procedure

$$[A|I] \xrightarrow{\text{row operations}} \left[I|A^{-1}\right]. \tag{33}$$

fails for matrix equations that are either inconsistent or overlapping.

Consider the pair of equations: $x_1 + x_2 = 2.0$ and $x_1 + x_2 = 1.0$. Applying row operations to the augmented form gives:

$$\left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right] \xrightarrow{r_2 \to r_2 - r_1} \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{array}\right]. \tag{34}$$

We conclude that $\left[A^{-1}\right]$ does not exist. The equations are either inconsistent or overlapping.

Matrix Rank

Matrix Rank

Definition. The rank of a matrix A, denoted rank(A), is the dimension of the vector space spanned by its rows (or columns).

- It is the number of linearly independent rows (or columns) in the matrix.

Theorem. For a $(n \times n)$ matrix A, the inverse A^{-1} exists $\iff$ $\text{rank(A)} = \text{n}$. Conversely, matrix A is singular if $\text{rank(A)} < \text{n}$ (i.e., rank deficient).

Computational Procedure. The standard way of determining the rank of a matrix is to:

- Transform the matrix to row echelon form.
- The rank is equal to the number of rows containing non-zero elements.

Matrix Rank

Example 1. The matrix

$$A = \begin{bmatrix} 3 & 1 & 9 \\ 1 & -2 & 5 \\ 2 & 3 & 4 \end{bmatrix}, \tag{35}$$

Applying row operations gives:

$$\begin{bmatrix} 3 & 1 & 9 \\ 1 & -2 & 5 \\ 2 & 3 & 4 \end{bmatrix} \xrightarrow{r_1 \to r_1 - r_2 - r_3} \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 5 \\ 2 & 3 & 4 \end{bmatrix}. \tag{36}$$

A has rank 2 because (by construction) row 1 is the sum of rows 2 and 3 (i.e., $row_1 - row_2 - row_3 = 0$).

Matrix Rank

Example 2. Let $x_1 + x_2 = 2.0$ and $x_1 + x_2 = 1.0$. Applying row operations to $[A|B]$ gives:

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 1 & 1 & 1 \end{array}\right] \xrightarrow{r_2 \to r_2 - r_1} \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 0 & -1 \end{array}\right]. \tag{37}$$

Inconsistent: $[A]$ is singular and rank $[A|B] \neq$ rank $[A]$.

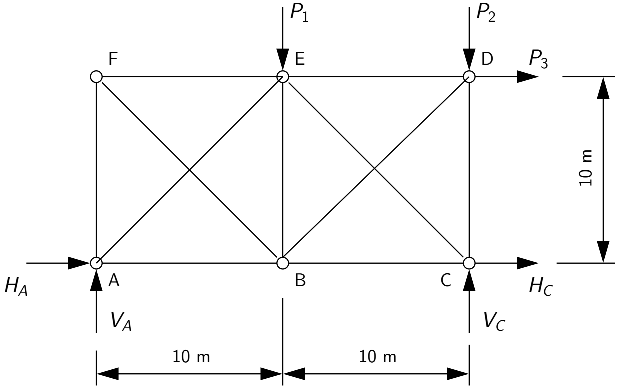
Example 3. Let $x_1 + x_2 = 2.0$ and $2x_1 + 2x_2 = 4.0$. Applying row operations to $[A|B]$ gives:

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 2 & 2 & 4 \end{array}\right] \xrightarrow{r_2 \to r_2 - 2r_1} \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 0 & 0 \end{array}\right]. \tag{38}$$

Overlapping: $[A]$ is singular and rank $[A|B]$ equals rank $[A]$.

Matrix Rank

Example 4. Consider equilibrium of the pin-jointed frame subject to external loads P_1, P_2 and P_3.



We wish to know the reactions as a function of applied forces.

Matrix Rank

Equations of equilibrium (not completely correct):

$$\sum H = 0 \to H_A + H_C + P_3 = 0.$$
$$\sum V = 0 \to V_A + V_C - P_1 - P_2 = 0.$$
$$\sum M_A = 0 \to -20V_C + 10P_1 + 20P_2 + 10P_3 = 0.$$
$$\sum M_C = 0 \to 20V_A - 10P_1 + 10P_3 = 0.$$

Writing the equations in matrix form:

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & -20 & 0 \\ 20 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_A \\ H_A \\ V_C \\ H_C \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ -1 & -1 & 0 \\ 10 & 20 & 10 \\ -10 & 0 & 10 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{39}$$

Matrix Rank

Symbolically, we have a set of matrix equations:

$$[A]\,[R] + [B]\,[P] = [0]\,. \tag{40}$$

If $\left[A^{-1}\right]$ exists, then we have:

$$[R] = -\left[A^{-1}\right][B]\,[P]\,. \tag{41}$$

Apply the following sequence of row operations:

- Scale row 3: $R_3 \rightarrow -R_3/20$
- Scale row 4: $R_4 \rightarrow R_4/20$
- Subtract rows 3 and 4 from row 2: $R_2 \rightarrow R_2 - R_3 - R_4$.
- Swap rows: $R_2 \longleftrightarrow R_4$, then $R_2 \longleftrightarrow R_1$.

Matrix Rank

Summary of Row Operations:

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & -20 & 0 \\ 20 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{row ops}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{42}$$

Conclusion:

- Because matrix A is rank deficient (i.e. rank(A) $= 3 < 4$), the matrix inverse $\left[A^{-1}\right]$ does not exist, and a unique solution to this problem cannot be found.
- The error lies in the use of $\sum M_A = 0$ and $\sum M_C = 0$ – they are not independent.

Summary of Results

Summary of Results

Classification of Solutions

The results are:

- A unique solution $\{X\} = \left[A^{-1}\right] \cdot \{B\}$ exists when $\left[A^{-1}\right]$ exists (i.e., $\det[A] \neq 0$).
- The equations are inconsistent when $[A]$ is singular and $\operatorname{rank}[A|B] \neq \operatorname{rank}[A]$.
- If $\operatorname{rank}[A|B]$ equals $\operatorname{rank}[A]$, then there are an infinite number of solutions.

Working Examples

Example 1. Numerical Solution of Matrix Equations

Problem Statement: Consider the matrix equations:

$$\begin{bmatrix} 3 & -6 & 7 \\ 9 & 0 & -5 \\ 5 & -8 & 6 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ -4 \end{bmatrix} \tag{43}$$

Theoretical Considerations:

- A unique solution $\{X\} = \left[A^{-1}\right] \cdot \{B\}$ exists when $\left[A^{-1}\right]$ exists (i.e., $\det[A] \neq 0$). Expanding det(A) about the first row gives:

$$\begin{aligned} det(A) &= 3\det\begin{bmatrix} 0 & -5 \\ -8 & 6 \end{bmatrix} + 6\det\begin{bmatrix} 9 & -5 \\ 5 & 6 \end{bmatrix} + 7\det\begin{bmatrix} 9 & 0 \\ 5 & -8 \end{bmatrix}, \\ &= 3(0-40) + 6(54+25) + 7(-72-0) = -150. \end{aligned} \tag{44}$$

Example 1. Numerical Solution of Matrix Equations

Program Source Code:

```
# ==================================================================
# TestMatrixEquations01.py: Compute solution to matrix equations.
#
# Written by: Mark Austin                           November 2022
# ==================================================================

import numpy as np
from numpy.linalg import matrix_rank

# Function to print two-dimensional matrices ...

def PrintMatrix(name, a):
    print("Matrix: {:s} ".format(name) );
    for row in a:
        for col in row:
            print("{:8.3f}".format(col), end=" ")
        print("")

# main method ...

def main():
    print("--- Enter TestMatrixEquations01.main()      ... ");
    print("--- ======================================== ... ");

    print("--- Part 1: Create test matrices ... ");
```

Example 1. Numerical Solution of Matrix Equations

Program Source Code: Continued ...

```
    A = np.array( [ [ 3, -6,  7],
                    [ 9,  0, -5],
                    [ 5, -8,  6] ]);
    PrintMatrix("A", A);

    B = np.array([ [3], [3], [-4] ]);
    PrintMatrix("B", B);

    print("--- Part 2: Check properties of matrix A ... ");

    rank  = matrix_rank(A)
    det   = np.linalg.det(A)

    print("--- Matrix A: rank = {:f}, det = {:f}  ...".format(rank, det) );

    print("--- Part 3: Solve A.x = B ... ");

    x = np.linalg.solve(A, B)
    PrintMatrix("x", x);

    print("--- ======================================= ... ");
    print("--- Leave TestMatrixEquations01.main()      ... ");

# call the main method ...

main()
```

Example 1. Numerical Solution of Matrix Equations

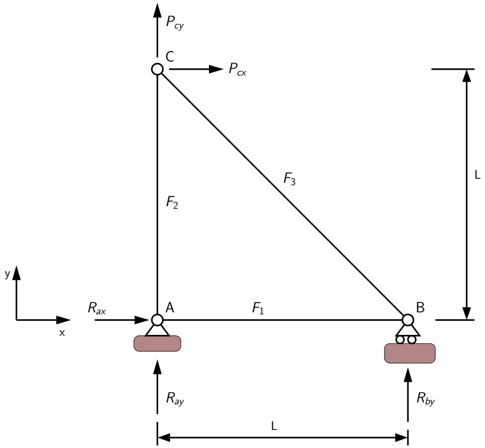
Abbreviated Output:

```
# Part 1: Create test matrices ...

Matrix: A
   3.000   -6.000    7.000
   9.000    0.000   -5.000
   5.000   -8.000    6.000

Matrix: B
   3.000
   3.000
  -4.000

# Part 2: Check properties of matrix A ...

Matrix A: rank = 3.000000, det = -150.000000  ...
```

```
# Part 3: Solve A.x = B ...

Matrix: x
   2.000
   4.000
   3.000
```

Example 2. Analysis of a Three-Bar Truss

Problem Statement: We wish to compute the support reactions and element-level forces in a three-bar truss:



Example 2. Analysis of a Three-Bar Truss

Equations of Equlibrium: Node A:

$$\sum F_{ax} = 0 \rightarrow F_1 + R_{ax} = 0, \sum F_{ay} = 0 \rightarrow F_2 + R_{ay} = 0. \quad (45)$$

Node B:

$$\sum F_{bx} = 0 \rightarrow F_1 + \frac{F_3}{\sqrt{2}} = 0, \sum F_{by} = 0 \rightarrow \frac{F_3}{\sqrt{2}} + R_{by} = 0. \quad (46)$$

Node C:

$$\sum F_{cx} = 0 \rightarrow \frac{-F_3}{\sqrt{2}} = P_{cx}, \sum F_{cy} = 0 \rightarrow F_2 + \frac{F_3}{\sqrt{2}} = P_{cy}. \quad (47)$$

Example 2. Analysis of a Three-Bar Truss

Program Source Code: ...

```
# ===================================================================
# TestTrussAnalysis01.py: Compute distribution of element forces
# and support reactions in a small three-bar truss.
#
# Written by: Mark Austin                            November 2023
# ===================================================================

import math
import numpy as np
from numpy.linalg import matrix_rank

# ===================================================================
# Function to print two-dimensional matrices ...
# ===================================================================

def PrintMatrix(name, a):
    print("Matrix: {:s} ".format(name) );
    for row in a:
        for col in row:
            print("{:8.4f}".format(col), end=" ")
        print("")

# ===================================================================
# main method ...
# ===================================================================

def main():
```

Example 2. Analysis of a Three-Bar Truss

Program Source Code: Continued ...

```
    print("--- Part 1: Initialize coefficients for matrix equations ... ");

    # Node A ...

    a11 =  1       # <--- equilibrium in x direction ...
    a14 =  1
    a22 =  1       # <--- equilibrium in y direction ...
    a25 =  1

    # Node B ...

    a31 = 1               # <--- equilibrium in x direction ...
    a33 = 1/math.sqrt(2)
    a43 = 1/math.sqrt(2) # <--- equilibrium in y direction ...
    a46 = 1

    # Node C ...

    a53 = -1/math.sqrt(2)  # <--- equilibrium in x direction ...
    a62 =  1               # <--- equilibrium in y direction ...
    a63 =  1/math.sqrt(2)

    # Load Vector B ...

    print("--- Part 2: Initialize load vector ... ");

    b5 = 10
    b6 =  0
```

Example 2. Analysis of a Three-Bar Truss

Program Source Code: Continued ...

```
    print("--- Part 3: Create test matrices ... ");

    A = np.array([ [ a11,   0,   0, a14,   0,   0 ],
                   [   0, a22,   0,   0, a25,   0 ],
                   [ a31,   0, a33,   0,   0,   0 ],
                   [   0,   0, a43,   0,   0, a46 ],
                   [   0,   0, a53,   0,   0,   0 ],
                   [   0, a62, a63,   0,   0,   0 ] ]);
    PrintMatrix("A", A);

    B = np.array([ [0], [0], [0], [0], [b5], [b6] ]);
    PrintMatrix("B", B);

    print("--- Part 4: Check properties of matrix A ... ");

    rank  = matrix_rank(A)
    det   = np.linalg.det(A)

    print("--- Matrix A: rank = {:f}, det = {:f}  ...".format(rank, det) );

    print("--- Part 5: Solve A.x = B ... ");

    x = np.linalg.solve(A, B)
    PrintMatrix("x", x);
```

Example 2. Analysis of a Three-Bar Truss

Program Source Code: Continued ...

```
    print("--- Part 6: Print support reactions and element-level forces ... ");

    print("--- Support A: R_ax = {:7.2f} ... ".format( x[3][0] ) );
    print("---           : R_ay = {:7.2f} ... ".format( x[4][0] ) );
    print("--- Support B: R_by = {:7.2f} ... ".format( x[5][0] ) );

    print("--- Element A-B: F1 = {:7.2f} ... ".format( x[0][0] ) );
    print("--- Element A-C: F2 = {:7.2f} ... ".format( x[1][0] ) );
    print("--- Element B-C: F3 = {:7.2f} ... ".format( x[2][0] ) );

# call the main method ...

main()
```

Example 2. Analysis of a Three-Bar Truss

Abbreviated Output:

```
--- Part 1: Initialize coefficients for matrix equations ...
--- Part 2: Initialize load vector ...
--- Part 3: Create test matrices ...

Matrix: A
  1.0000    0.0000    0.0000    1.0000    0.0000    0.0000
  0.0000    1.0000    0.0000    0.0000    1.0000    0.0000
  1.0000    0.0000    0.7071    0.0000    0.0000    0.0000
  0.0000    0.0000    0.7071    0.0000    0.0000    1.0000
  0.0000    0.0000   -0.7071    0.0000    0.0000    0.0000
  0.0000    1.0000    0.7071    0.0000    0.0000    0.0000

Matrix: B
  0.0000
  0.0000
  0.0000
  0.0000
 10.0000
  0.0000
```

Example 2. Analysis of a Three-Bar Truss

Abbreviated Output: Continued ...

```
--- Part 4: Check properties of matrix A ...
--- Matrix A: rank = 6.000000, det = 0.707107  ...
--- Part 5: Solve A.x = B ...

Matrix: x
 10.0000
 10.0000
-14.1421
-10.0000
-10.0000
 10.0000

--- Part 6: Print support reactions and element-level forces ...

--- Support A: R_ax =  -10.00       --- Element A-B: F1 =   10.00
---          : R_ay =  -10.00       --- Element A-C: F2 =   10.00
--- Support B: R_by =   10.00       --- Element B-C: F3 =  -14.14
```

Example 3. Symbolic Computation of Matrix Determinant

Problem Statement: Determinant of (2x2) matrix:

$$det(A) = det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a \cdot d - b \cdot c. \tag{48}$$

Determinant of (3x3) matrix:

$$det(A) = det \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = aM_{11} - bM_{12} + cM_{13}. \tag{49}$$

where,

$$M_{11} = \begin{bmatrix} e & f \\ h & i \end{bmatrix}, M_{12} = \begin{bmatrix} d & f \\ g & i \end{bmatrix}, M_{13} = \begin{bmatrix} d & e \\ g & h \end{bmatrix}. \tag{50}$$

Example 3. Symbolic Computation of Matrix Determinant

Program Source Code: ...

```
# ==================================================================================
# TestMatrixDeterminant01.py: Compute symbolic description of matrix determinant
# ==================================================================================

import sympy as sp
from sympy import Integral, Matrix, pi, pprint

# main method ...

def main():
    print("--- Part 1: Determinant of 2x2 symbolic matrix ...");

    a, b, c, d = sp.symbols('a,b,c,d')
    arr1 = sp.Matrix(( [a,b], [c,d] ))

    pprint(arr1)

    print("--- Compute and print matrix determinant ...");
    print("--- arr1.det() --> {:s} ...".format( str( sp.det(arr1) ) ))

    print("--- Set values: a = 1, b = 2, c = 3, d = 4 ...");
    print("--- arr1 = arr1.subs( {a:1, b:2, c:3, d:4} ) \n")
    arr1 = arr1.subs( {a:1, b:2, c:3, d:4} )

    pprint(arr1)
    print("--- arr1.det() = {:s} ...".format( str( sp.det(arr1) ) ))
```

Example 3. Symbolic Computation of Matrix Determinant

Program Source Code: Continued ...

```
    print("--- Part 2: Determinant of 3x3 symbolic matrix ...");

    a, b, c, d, e, f, g, h, i = sp.symbols('a,b,c,d,e,f,g,h,i')

    arr2 = sp.Matrix(( [a,b,c], [d,e,f], [g,h,i] ))
    pprint(arr2)

    print("--- Compute and print matrix determinant ...");
    print("--- arr2.det() --> {:s} ...".format( str( sp.det(arr2) ) ))

    print("--- Set values in 2nd/3rd rows: d = 4, ..., i = 9 ...");
    print("--- arr2 = arr2.subs( {d:4, e:5, f:6, g:7, h:8, i:9} ) \n")
    arr2 = arr2.subs( {d:4, e:5, f:6, g:7, h:8, i:9 } )

    pprint(arr2)
    print("--- arr2.det() = {:s} ...".format( str( sp.det(arr2) ) ))

    print("--- Set values in 1st row: a = 4, b = 2, c = 3 ...");
    print("--- arr2 = arr2.subs( {a:1, b:2, c:3} ) \n")
    arr2 = arr2.subs( { a:1, b:2, c:3 } )

    pprint(arr2)
    print("--- arr2.det() = {:s} ...".format( str( sp.det(arr2) ) ))

# call the main method ...

main()
```

Example 3. Symbolic Computation of Matrix Determinant

Abbreviated Output: Part 1

```
--- Part 1: Determinant of 2x2 symbolic matrix ...

| a  b |
|      |
| c  d |

--- Compute and print matrix determinant ...
--- arr1.det() --> a*d - b*c ...

--- Set values: a = 1, b = 2, c = 3, d = 4 ...
--- arr1 = arr1.subs( {a:1, b:2, c:3, d:4} )

| 1  2 |
|      |
| 3  4 |

--- arr1.det() = -2 ...
```

Example 3. Symbolic Computation of Matrix Determinant

Abbreviated Output: Part 2

```
--- Part 2: Determinant of 3x3 symbolic matrix ...

| a  b  c |
|         |
| d  e  f |
|         |
| g  h  i |

--- Compute and print matrix determinant ...
--- arr2.det() --> a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g ...

--- Set values in 2nd/3rd rows: d = 4, ..., i = 9 ...
--- arr2 = arr2.subs( {d:4, e:5, f:6, g:7, h:8, i:9} )

| a  b  c |
|         |
| 4  5  6 |
|         |
| 7  8  9 |

--- arr2.det() = -3*a + 6*b - 3*c ...
```

Example 4. Symbolic Solution to Matrix Equations

Problem Statement: We will use sympy, Python's symbolic processing module, to compute algebraic solutions to the (2x2) matrix equations:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} e \\ f \end{bmatrix}. \tag{51}$$

and (3x3) matrix equations:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} j \\ k \\ l \end{bmatrix}. \tag{52}$$

Then, we will compute solutions to specific problems by assigning numerical values to the matrix elements.

Example 4. Symbolic Solution to Matrix Equations

Program Source Code: ...

```
# ======================================================================================
# TestMatrixSolutions01.py: Compute symbolic solutions to linear matrix equations ...
# ======================================================================================

import sympy as sp
from sympy import Integral, Matrix, pi, pprint

# main method ...

def main():
    # Define symbolic representation of matrix ...

    print("--- Part 1: Solution of 2x2 symbolic matrix equations ...");

    a, b, c, d, e, f = sp.symbols('a,b,c,d,e,f')
    arr1 = sp.Matrix(( [a,b], [c,d] ))
    b1   = sp.Matrix(( [e], [f] ))

    pprint(arr1)
    pprint(b1)

    print("--- General symbolic solution to (2x2) matrix system ...\n");

    solution = arr1.solve(b1)
    print(solution)

    print("--- Expressions for individual solution elements ...\n");
```

Example 4. Symbolic Solution to Matrix Equations

Program Source Code: Continued ...

```
    print("--- x = {:s} ...".format( str( solution[0] ) ))
    print("--- y = {:s} ...".format( str( solution[1] ) ))

    print("--- Set values in arr1: a =  1, b =  2, c = 3, d = 4 ...");
    print("--- Set values in   b1: e = 25, f = 55 ...");

    arr1 = arr1.subs( {a:1, b:2, c:3, d:4} )
    b1   =   b1.subs( {e:25, f:55} )

    print("--- Check matrix rank ...\n");
    print("--- arr1.rank() --> {:d} ...".format( arr1.rank() ) );

    print("--- Evaluate symbolic solution expressions ...\n");

    solution = arr1.solve(b1)
    print(solution)

    print("--- x = {:s} ...".format( str( solution[0] ) ))
    print("--- y = {:s} ...".format( str( solution[1] ) ))

    print("--- Part 2: Solution of 3x3 symbolic matrix equations ...");

    a, b, c, d, e, f, g, h, i, j, k, l = sp.symbols('a,b,c,d,e,f,g,h,i,j,k,l')
    arr2 = sp.Matrix(( [a,b,c], [d,e,f], [g,h,i] ))
    b2   = sp.Matrix(( [j], [k], [l] ))

    pprint(arr2)
    pprint(b2)
```

Example 4. Symbolic Solution to Matrix Equations

Program Source Code: Continued ...

```
    print("--- General symbolic solution to (2x2) matrix system ...\n");

    solution = arr2.solve(b2)
    print(solution)

    print("--- Expressions for individual solution elements ...\n");
    print("--- x1 = {:s} ...".format( str( solution[0] ) ))
    print("--- x2 = {:s} ...".format( str( solution[1] ) ))
    print("--- x3 = {:s} ...".format( str( solution[2] ) ))

    print("--- Set values in arr2 ...");
    print("---   arr2 = arr2.subs( {a:3, b:-6, c:7, d:9, e:0, f:-5, g:5, h:-8, i:6 } ) .
    print("---     b2 =   b2.subs( {j:3, k:-6, l:7} ) ...\n")

    arr2 = arr2.subs( {a:3, b:-6, c:7, d:9, e:0, f:-5, g:5, h:-8, i:6 } )
    b2   =   b2.subs( {j:3, k:3, l:-4} )

    pprint(arr2)
    pprint(b2)

    print("\n--- Check matrix rank and determinant ...\n");
    print("--- arr2.rank() --> {:d} ...".format( arr2.rank() ) );
    print("--- arr2.det()  --> {:s} ...".format( str( arr2.det() )) );

    print("--- Evaluate symbolic solution expressions ...\n");

    solution = arr2.solve(b2)
    print(solution)
```

Example 4. Symbolic Solution to Matrix Equations

Abbreviated Output: Part 1

```
--- Part 1: Solution of 2x2 symbolic matrix equations ...

    | a   b |          | e |
    |       |          |   |
    | c   d |          | f |

--- Expressions for individual solution elements ...

--- x = (-b*f + d*e)/(a*d - b*c) ...
--- y = (a*f - c*e)/(a*d - b*c) ...

--- Set values in arr1: a =  1, b =  2, c = 3, d = 4 ...
--- Set values in   b1: e = 25, f = 55 ...
--- arr1.rank() --> 2 ...

--- Evaluate symbolic solution expressions ...

--- x = 5, y = 10 ...
```

Example 4. Symbolic Solution to Matrix Equations

Abbreviated Output: Part 2

```
--- Part 2: Solution of 3x3 symbolic matrix equations ...

    | a  b  c |          | j |
    |         |          |   |
    | d  e  f |          | k |
    |         |          |   |
    | g  h  i |          | l |

--- Expressions for individual solution elements ...

--- x1 = (b*f*l - b*i*k - c*e*l + c*h*k + e*i*j - f*h*j)/
         (a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g) ...

--- x2 = (-a*f*l + a*i*k + c*d*l - c*g*k - d*i*j + f*g*j)/
         (a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g) ...

--- x3 = (a*e*l - a*h*k - b*d*l + b*g*k + d*h*j - e*g*j)/
         (a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g) ...
```

Example 4. Symbolic Solution to Matrix Equations

Abbreviated Output: Part 2 continued ...

```
--- Set values in arr2 ...

---   arr2 = arr2.subs( {a:3, b:-6, c:7, d:9, e:0, f:-5, g:5, h:-8, i:6 } ) ...
---     b2 =   b2.subs( {j:3, k:-6, l:7} ) ...

    | 3  -6   7 |          |  3 |
    |           |          |    |
    | 9   0  -5 |          |  3 |
    |           |          |    |
    | 5  -8   6 |          | -4 |

--- Check matrix rank and determinant ...

--- arr2.rank() --> 3 ...
--- arr2.det()  --> -150 ...

--- Evaluate symbolic solution expressions ...

--- x1 = 2 ...
--- x2 = 4 ...
--- x3 = 3 ...
```