| **S No** | **Method Name** | **Benifits** | **Limitations** |
| --- | --- | --- | --- |
| 1 | Graphical Method | This graphical method provides a visual representation of the system of equations. It is suitable for two variables equations on a two-dimensional plane. | 1. impractical for systems with more than two variables.  2. Difficult to visualize and plot equations in higher dimensions.  3. They may not yield precise values unless the graph is carefully constructed.  4. Infeasible for non-linear or complex equations |
| 2 | Substitution Method | 1. It is relatively easy to understand and implement.  2. It follows a systematic step-by-step process to track every step. | 1. There is chance make mistakes when dealing with multiple variables and equations.  2. It is not effective for systems with non-linear equations.  3. It has a limited effeciency when dealing with systems with a large number of equations. |
| 3 | Elimination Method | 1. Simplicity  2. Versatility  3. Exact Solutions  4. Conceptual Understanding | 1. Complexity with non-linear equations  2. Time-consuming for large systems  3. Potential for errors  4. Dependency on initial equations |
| 4 | Matrix Method | 1. Compact Representation  2. Efficient Computation  3. Solution Existance and Uniquness  4. Scalibility | 1. Complexity with Non-linear Equations  2. Matrix Inversion Difficulties  3. Sensitivity to Rounding Errors  4. Conceptual Understanding |
| 5 | Gaussian Elimination Method | 1. Systematic and algorithmic  2. Exact Solutions  3. Versatility  4. Efficiency | 1. Complexity with non-linear equations  2. Sensitivity to rounding errors  3. Dependency on initial equations  4. Computational complexity for large systems |
| 6 | Gauss-Jordon Elimination Method | 1. Exact Solutions  2. Uniqueness of Solutions  3. Diagonal Matrix Form  4. Elimination of Free Variables | 1. Complexity with non-linear equations  2. Sensitivity to rounding errors  3. Computational complexity for large systems  4. Dependency on initial equations |
| 7 | LU (Lower Upper) Decomposition Method | 1. Efficiency  2. Numerical Stability  3. Inverse and Determinant Computation  4. Matrix Inversion | 1. Applicability to square matrices  2. Lack of existence or uniqueness  3. Additional storage requirements  4. Sensitivity to pivoting |
| 8 | SVD (Singular Value Decomposition Method) Method | 1. Dimensionality reduction  2. Robustness to noise  3. Data compression  4. Pseudoinverse computation | 1. Computational complexity  2. Interpretability  3. Lack of uniqueness  4. Sensitivity to scaling |
| 9 | Iterative Method | 1. Efficiency for large-scale problems  2. Flexibility and adaptability  3. Convergence monitoring  4. Incremental and online computation | 1. Convergence rate and speed  2. Sensitivity to initial guess  3. Iteration termination criteria  4. Dependence on problem characteristics |
| 10 | Cramer's Rule | 1. Simplicity and clarity  2. Exact solutions  3. Independence of equations | 1. Computational inefficiency  2. Sensitivity to matrix properties  3. Numerical stability  4. Limited applicability |

The below table has been copied from day-48 lecture “Advance Methods”

| **Method** | **Typical Equation** | **Steps to Resolve** | **Limitations** | **Benefits** |
| --- | --- | --- | --- | --- |
| Graphical Method | y = mx + c | Plot each equation and find intersection points. | Impractical for more than 2 variables; accuracy depends on scale. | Intuitive and visual; good for understanding the nature of solutions. |
| Substitution Method | x + y = b | Solve one equation for a variable, substitute it into others, and solve. | Can be cumbersome for complex systems. | Simple and straightforward for small systems. |
| Elimination Method | ax + by = c | Add or subtract equations to eliminate a variable, then solve for others. | Can get complex with many variables. | Effective for linear equations; straightforward for small systems. |
| Matrix Method (Inversion) | Ax = B | Formulate matrix equation, calculate inverse of A, compute A-1B. | Infeasible for non-square or singular matrices. | Systematic and precise; good for complex systems. |
| Gaussian Elimination | Ax = B | Convert to upper triangular form using row operations, then back substitute. | Can be computationally intensive for large matrices. | General method, applicable to most systems. |
| Gauss-Jordan Elimination | Ax = B | Reduce matrix to row echelon form, directly read off solutions. | Similar to Gaussian; can be computationally intensive. | Simplifies to a direct solution without back substitution. |
| LU Decomposition | Ax = B | Decompose A into LU, solve Ly = B and then Ux = y. | Requires additional steps to perform decomposition. | Efficient for multiple systems with the same A. |
| Singular Value Decomposition | Ax = B | Decompose A into U, Σ, V, use these to solve the system. | Complex and requires understanding of advanced linear algebra. | Powerful in data science and for ill-conditioned systems. |
| Iterative Methods | Ax = B | Start with a guess, iteratively refine the solution. | Convergence can be slow; not always guaranteed. | Useful for very large systems where direct methods fail. |
| Cramer’s Rule | ax + by = c | Use determinants to solve, each variable calculated separately. | Only for square matrices with non-zero determinants. | Straightforward for small systems; provides direct solution. |