Exploring methods for solving linear systems

By Kevin Perez

Abstract

This document provides an introduction to direct methods for solving linear systems of equations for an audience interested in computationally implementing and solving linear systems. It may be used as a reference by engineers that are seeking the best method for solving a problem that can be described as a linear system. An application of linear algebra in engineering is explored to motivate the theme. Key topics in linear algebra are highlighted and briefly introduced. A randomly generated linear system with a square coefficient matrix is thoroughly analyzed and solved while introducing the direct methods.

1. Introduction

Rationale

Modern real world processes described by systems of linear equations tend to be overwhelmingly complex because of the vast amount of data that is available. A linear system generated from some structured or unstructured data source of unbeknown size may need to be quickly solved because of business time constraints or other critical requirements. To consistently solve linear systems quickly, a computer or computer operator should know which of the well-established methods to apply. This project introduces some of the most used methods for solving a linear system with the perspective of a novice in the field of linear algebra. This project will reinforce existing knowledge about linear systems and assure the reliability of existing methods for solving linear systems.

Purpose

The purpose of this project is to document the capabilities of linear system solvers and assure the reliability of each method. The information is delivered from the perspective of novices in the field of linear algebra, and with this regard, the reader should consider the information within is easy to digest and straightforward. The goal of the writers is to introduce and motivate linear algebra as a powerful tool for solving system characterized by linear equations. To readers with prior knowledge of the subject, the content below will reinforce existing knowledge about linear systems.

Research questions

Can a linear system retain its integrity after decomposition? Will any combination of valid row operations really generate a row equivalent system with different scalar values?

Learning the advantages and disadvantages of all linear solvers is important in order to decide which is the most appropriate and optimal method for solving systems of linear equations, or linear systems. There are methods specifically designed for solving linear systems that fit certain properties such as the dimensions of the matrices and vectors and whether there is a pattern in the dispersion of coefficients in the matrix. In this document, a linear system defined over the field of real numbers will be dissected and solved using several methods to prove the capability and reliability of the solutions of each method provides.

2. Linear system concepts in engineering and science

ISO/IEC 11172-3:1993: The MPEG-1 Standard

When vectors are introduced in a Calculus course, they are described as having magnitude and direction. However, vectors can be used to model abstract objects such as audio signals where the components of the vectors represent a frequencies defined by a periodic function.

A problem faced in audio signal processing is encoding analog sound signals into PCM (Pulse Code Modulation) sound samples and then compressing the samples to MPEG-1 Audio Layer 1 (or MP1) defined in ISO/IEC 11172-3. The analog to digital conversion process which involves measuring voltages in fixed time intervals is called sampling. The amount of measurements per time interval is the *sampling rate*. Each sample is stored in a fixed amount of bits. To put into perspective the amount of signals that are processed, a Digital Audio Compact Disc is encoded with a stereo (2-channel) audio signal where each channel is sampled at 44.1kHz with 16 bits per sample. That is, analog signals are encoded into 1,411,200 bits per second!

$$\mathrm{DCT}_6{}^{\mathrm{III}} = \begin{pmatrix} \cos(0) & \cos(\pi/12) & \cos(\pi/6) & \cos(\pi/4) & \cos(\pi/3) & \cos(5\pi/12) \\ \cos(0) & \cos(\pi/4) & \cos(\pi/2) & \cos(3\pi/4) & \cos(\pi) & \cos(5\pi/4) \\ \cos(0) & \cos(5\pi/12) & \cos(5\pi/6) & \cos(5\pi/4) & \cos(5\pi/3) & \cos(25\pi/12) \\ \cos(0) & \cos(7\pi/12) & \cos(7\pi/6) & \cos(7\pi/4) & \cos(7\pi/3) & \cos(35\pi/12) \\ \cos(0) & \cos(3\pi/4) & \cos(3\pi/2) & \cos(9\pi/4) & \cos(3\pi) & \cos(15\pi/4) \\ \cos(0) & \cos(11\pi/12) & \cos(11\pi/6) & \cos(11\pi/4) & \cos(11\pi/3) & \cos(55\pi/12) \end{pmatrix}$$

The matrix above represents the fast, discrete, type 3, cosine transform derived from the following formula: $\mathrm{DCT}_6{}^{\mathrm{III}}=(\cos((\pi/\mathrm{n})(i+1/2)k))$ for $i=0$ to n-1 and $k=0$ to n-1 [3, pp. 10]. It is used to synthesize sequences of (PCM) subband samples where each subband sample contains quantized values of a narrow spectrum of sound frequency [3, pp. 6]. The matrix above, smaller than the typical matrix an encoder processes, shows expressions in each row that define a periodic function with a fixed frequency. The matrix can be used to transform other matrices or vectors . It should also be pointed out that the matrix can be reduced prior to transforming a signal because there are repeating wave amplitudes. Other cosine transform matrices can be used if the granularity of the audio samples needs to be increased.

The MPEG-1 standard specifies only the bitstream and decoding process, but the encoder can be implemented differently. This is a good example for why engineers need to be familiar with linear algebra and how to programmatically implement methods for solving linear systems.

3. Problem statement and system description

This section will introduce some common terms and symbols used throughout this document that will help readers digest the material discussed in subsequent sections.

Below is the system of linear equations that we will be solving. We need to determine whether the b

can be written as a linear combination of a_1, a_2,..., a_{20}.

$$\begin{cases} a_{11}\,x_1 + \cdots + a_{1\,j}\,x_j + \cdots + a_{1\,m}\,x_m = b_1 \\ \vdots \\ a_{i\,1}\,x_1 + \cdots + a_{ij}\,x_j + \cdots + a_{im}\,x_m = b_i \\ \vdots \\ a_{n\,1}\,x_1 + \cdots + a_{nj}\,x_j + \cdots + a_{nm}\,x_m = b_n \end{cases}$$

This **system of linear equations**, denoted by the left curly brace emphasizing the relationship among the equations, known as **vector equations**, can be converted to a **matrix equation** of the form Ax=b.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1\,m} \\ a_{21} & a_{22} & \cdots & a_{2\,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n\,1} & a_{n\,2} & \cdots & a_{nm} \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

A is referred to as the coefficient matrix with n rows and m columns. It contains only the coefficients from the expressions on the left hand side of the equations. In the case that all coefficients are real values, A is said to be a real matrix, or $A \in \mathbb{R}^{n\,x\,m}$. If the matrix the same number of rows and columns, then it is called a square matrix.

The expressions on the right hand side of the equations are assigned to the rows of a **column vector** denoted by the letter $\mathbf{b} \in \mathbb{R}^n$.

The letter $\mathbf{x} \in \mathbb{R}^m$ denotes the column vector of unknown variables that should satisfy all equations in the system simultaneously. These are the variables in question when solving a system of linear equations. More than one vector **x** may satisfy the system.

Variables A, **b**, and **x** and their variations will be used throughout this document to succinctly represent a linear system.

Viewing a linear system in the matrix format is useful because it detracts from using the variables x_1,x_2, ... , x_n and equality signs in each equation while operating on the numerical values of the system. Viewing only the numerical values in the system makes it easier to transform the system in order to arrive at a solution, or solutions, if any exist. The abstraction is also a natural representation in computer programs such as Mathematica because values are stored in a data structure similar to a matrix. Therefore the matrix format will be used from this point forward to represent and solve the system.

When solving equations, the expressions on both sides of the equality sign need to receive the same operation in order for the equation to remain symmetrical. To enforce the same rules on a linear system an **augmented matrix** is formed. An $n\,x\,m$ augmented matrix, denoted $(A \mid \mathbf{b})$, which contains the columns of A in the first (n-1) columns and the column vector **b** as column n is a useful representation of the system because it serves as a reminder that the same operations performed on the left hand side of the equation must be performed on the right hand side due to the symmetry of equality. This will be a useful form for solving a linear system using particular methods.

$$A_{aug} = (A \mid \mathbf{b}) = \left(\begin{array}{cccc:c} a_{11} & a_{12} & \cdots & a_{1\,n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2\,n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m\,1} & a_{m\,2} & \cdots & a_{mn} & b_m \end{array}\right)$$

The vertical line in the augmented matrix separating the values of A and **b** is usually omitted when

solving a system in this form.

4. Linear equations declaration

To explore the methods for solving a linear system, a 20 by 20 real linear system has been generated. All steps in the process of solving the system have been accomplished with Wolfram Mathematica. Here is the initialization of the system of linear equations we will use in our demonstration.

```
(* Clearing variables to be used as the unknown variables *)
Clear[x1, x2, x3, x4, x5, x6, x7, x8, x9,
 x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20]
(* Initializing a list containing conventionally named
 variables that will be used to store the solutions
 represented by the unknown variables in the linear system. *)
vars = {x1, x2, x3, x4, x5, x6, x7, x8, x9, x10,
   x11, x12, x13, x14, x15, x16, x17, x18, x19, x20};

(* Initialize variables storing linear system equations *)
eq1 = 65.94 x1 - 182.45 x10 - 266.27 x11 + 462.43 x12 + 842.9 x13 + 233.29 x14 + 995.66 x15 +
     704.14 x16 + 148.03 x17 + 970.26 x18 + 371.37 x19 - 440.77 x2 - 188.38 x20 - 594.7 x3 +
     699.79 x4 - 417.02 x5 + 432.28 x6 - 572.34 x7 - 907.73 x8 - 267.13 x9 == 34.47 ;
eq2 = 626.79 x1 - 28.04 x10 + 807.26 x11 - 314.4 x12 + 297.48 x13 - 562.44 x14 - 238.82 x15 +
     926.47 x16 - 983.03 x17 - 613.3 x18 - 971.89 x19 + 632.85 x2 - 322.12 x20 + 832.5 x3 +
     919.47 x4 - 147.94 x5 + 129.16 x6 - 413.52 x7 - 903.96 x8 - 984.56 x9 == 903.16 ;
eq3 = 379.34 x1 - 410.84 x10 + 353.19 x11 + 500.78 x12 - 556.59 x13 - 192.94 x14 - 48.36 x15 +
     174.16 x16 + 784.86 x17 + 740.45 x18 + 767.39 x19 + 443.73 x2 - 351.58 x20 - 757.34 x3 +
     537.64 x4 - 531.19 x5 + 709.44 x6 + 353.84 x7 + 119.51 x8 + 271.98 x9 == -836.4 ;
eq4 = 469.02 x1 + 251.69 x10 - 571.51 x11 - 828.33 x12 - 472.26 x13 - 172.72 x14 +
     2.37 x15 - 218.71 x16 + 266.92 x17 - 953.73 x18 - 97.48 x19 + 668.92 x2 - 67.55 x20 -
     11.59 x3 + 753.51 x4 + 707.9 x5 + 840.63 x6 + 721.59 x7 - 481.73 x8 + 49.17 x9 == 317.83 ;
eq5 = 169.26 x1 - 78.5 x10 + 320.69 x11 - 98.82 x12 - 576.33 x13 + 157.87 x14 - 772.42 x15 +
     552.54 x16 + 304.47 x17 - 788.61 x18 - 182.78 x19 + 798.32 x2 - 826.79 x20 - 135.73 x3 -
     458.94 x4 + 40.17 x5 - 594.19 x6 - 334.26 x7 - 489.39 x8 + 808.86 x9 == -402. ;
eq6 = -552.42 x1 - 630.9 x10 + 631.84 x11 + 649.54 x12 + 133.88 x13 + 199.25 x14 +
     950.14 x15 - 377.78 x16 - 270.66 x17 - 775.68 x18 + 23.61 x19 - 113.19 x2 + 306.48 x20 -
     580.43 x3 + 990.18 x4 - 615.94 x5 + 718.8 x6 - 648.02 x7 - 826.76 x8 + 249. x9 == 790. ;
eq7 = -543.17 x1 + 68.47 x10 + 203.12 x11 - 59.59 x12 - 492.75 x13 + 860.32 x14 - 838.9 x15 +
     628.1 x16 - 9.23 x17 - 958.24 x18 + 850.51 x19 + 117.75 x2 - 62.33 x20 + 359.13 x3 +
     883.1 x4 + 865.15 x5 + 678.63 x6 - 238.36 x7 - 34.73 x8 + 52.67 x9 == -745.84 ;
eq8 = 811.41 x1 + 175.22 x10 - 860.87 x11 + 966.07 x12 - 352.11 x13 + 183.6 x14 -
```

```
    933.21 x15 + 370.88 x16 – 24.71 x17 + 380.07 x18 – 670.33 x19 + 12.92 x2 + 169.47 x20 –
    866.6 x3 + 19.57 x4 – 363.81 x5 – 875.38 x6 – 971.07 x7 + 845.35 x8 – 992.7 x9 == –903.4 ;
eq9 = –600.08 x1 – 356.16 x10 + 123.8 x11 – 470.64 x12 + 80.95 x13 – 387.57 x14 – 777.32 x15 –
    71.62 x16 – 403.13 x17 + 519.81 x18 + 516.76 x19 – 512.71 x2 + 186.9 x20 – 462.16 x3 +
    167.76 x4 + 630.07 x5 + 523.52 x6 – 335.63 x7 + 15.39 x8 – 936.2 x9 == –367.55 ;
eq10 = 378.95 x1 + 427.84 x10 – 807.15 x11 – 849.86 x12 – 571.91 x13 – 99.13 x14 – 414.3 x15 +
    645.45 x16 – 543.21 x17 – 35.32 x18 – 97.47 x19 – 467.7 x2 – 324.59 x20 – 861.62 x3 –
    111.97 x4 + 220.49 x5 + 170.97 x6 – 533.27 x7 – 658.91 x8 – 979.97 x9 == –564.15 ;
eq11 = –492.87 x1 + 592.74 x10 – 110.29 x11 + 362.36 x12 – 128.01 x13 –
    377.34 x14 + 638.49 x15 – 873.77 x16 – 928.8 x17 – 735.66 x18 +
    882.01 x19 + 227.51 x2 – 338.56 x20 – 667.17 x3 + 246.83 x4 +
    557.13 x5 – 856.95 x6 + 56.34 x7 + 798.44 x8 + 507.54 x9 == 278.43 ;
eq12 = 631.87 x1 – 619.1 x10 – 712.02 x11 – 454.88 x12 + 328.99 x13 + 839.72 x14 –
    51.3 x15 + 648.65 x16 – 4.9 x17 + 629.45 x18 – 539.45 x19 – 540.45 x2 – 666.51 x20 –
    238.7 x3 + 985.6 x4 + 152.46 x5 – 638.94 x6 – 398.08 x7 + 927.71 x8 – 136.97 x9 == 483. ;
eq13 = –284.08 x1 + 639.09 x10 – 848.11 x11 – 724.28 x12 + 494.36 x13 –
    698.93 x14 – 247.34 x15 + 286.78 x16 + 177.74 x17 – 158.99 x18 +
    854.85 x19 – 965.25 x2 – 643.42 x20 + 692.73 x3 + 435.32 x4 –
    421.15 x5 – 477.05 x6 – 778.96 x7 + 858.4 x8 – 583.6 x9 == –483.35 ;
eq14 = –696.29 x1 – 989.77 x10 – 465.1 x11 – 135.54 x12 – 210.06 x13 – 54.58 x14 –
    515.13 x15 + 943.77 x16 + 199.82 x17 – 119.55 x18 – 667.3 x19 – 353.8 x2 + 399.16 x20 –
    114.64 x3 + 531.11 x4 – 853.14 x5 – 501.3 x6 – 0.28 x7 + 92.94 x8 + 797.43 x9 == –197.66 ;
eq15 = 562.68 x1 – 793.07 x10 + 84.5 x11 + 223.55 x12 + 821.56 x13 + 744.73 x14 – 94.48 x15 –
    815.9 x16 + 188.7 x17 – 932.63 x18 + 174.47 x19 – 843.81 x2 – 969.09 x20 + 388.52 x3 –
    996.41 x4 – 96.69 x5 + 835.98 x6 + 774.89 x7 + 838.07 x8 + 827.36 x9 == 19.54 ;
eq16 = 204.75 x1 – 434.22 x10 + 171.11 x11 + 30.25 x12 – 879.34 x13 +
    826.11 x14 + 762.62 x15 + 611.79 x16 – 947.76 x17 + 269.34 x18 –
    266.05 x19 + 410.75 x2 – 843.78 x20 + 695.21 x3 + 136.15 x4 +
    159.5 x5 + 354.08 x6 + 113.38 x7 + 518.57 x8 + 583.42 x9 == 808.97 ;
eq17 = –686.75 x1 + 362.37 x10 – 271.52 x11 – 604.26 x12 – 577.72 x13 –
    281.44 x14 – 421.57 x15 – 318.52 x16 + 708.66 x17 – 802.64 x18 +
    139.21 x19 + 800.09 x2 + 180.08 x20 + 546.7 x3 – 70.72 x4 + 553.74 x5 –
    643.98 x6 – 111.55 x7 + 296.34 x8 – 864.56 x9 == –335.67 ;
eq18 = –737.56 x1 – 878.05 x10 + 606.05 x11 + 815.06 x12 – 34.63 x13 +
    234.29 x14 + 887.2 x15 + 322.67 x16 – 587.67 x17 + 964.14 x18 +
    348.76 x19 – 62.46 x2 – 918.05 x20 + 917.99 x3 + 840.56 x4 +
    819.04 x5 + 454.84 x6 – 819.73 x7 + 934.08 x8 – 535.84 x9 == –467.79 ;
eq19 = 927.55 x1 – 425.99 x10 – 814.86 x11 – 98.2 x12 – 671.63 x13 + 856.94 x14 + 749.07 x15 –
    191.32 x16 – 979.13 x17 + 923.73 x18 – 428.21 x19 + 77.85 x2 – 862.77 x20 – 959.3 x3 +
    941.95 x4 + 947.41 x5 + 895.7 x6 – 313.83 x7 – 597.41 x8 + 349.24 x9 == –232.68 ;
eq20 = –525.57 x1 – 670.52 x10 – 641.16 x11 + 614.21 x12 – 669.64 x13 –
    623.88 x14 – 819.64 x15 + 816.26 x16 + 905.78 x17 + 753.09 x18 –
    607.54 x19 – 614.14 x2 + 720.61 x20 – 906.27 x3 + 130.58 x4 +
```



```
    387.06 x5 - 402.56 x6 - 72.54 x7 - 421.9 x8 + 958.16 x9 == 786.12;

(* Combine equations to form linear system *)
linearSystem = {eq1, eq2, eq3, eq4, eq5, eq6, eq7, eq8, eq9,
   eq10, eq11, eq12, eq13, eq14, eq15, eq16, eq17, eq18, eq19, eq20};

(* Extract the coefficients in the system *)
linearSystemCoeff = Normal@CoefficientArrays[linearSystem, vars];

(* Store RHS coefficients of equations in a variable *)
b = -linearSystemCoeff[[1]];

(*Store LHS coefficients of equations in a variable *)
A = linearSystemCoeff[[2]];

originalSystem = matrixToSystem[A, b];

Style[DisplayForm[GridBox[{{"{", Column[originalSystem]}}]], 5]
```

$$\begin{bmatrix}
65.94 x_1 - 440.77 x_2 - 594.7 x_3 + 699.79 x_4 - 417.02 x_5 + 432.28 x_6 - 572.34 x_7 - 907.73 x_8 - 267.13 x_9 - 182.45 x_{10} - 266.27 x_{11} + 462.43 x_{12} + 842.9 x_{13} + 233.29 x_{14} + 995.66 x_{15} + 704.14 x_{16} + 148.03 x_{17} + 970.26 x_{18} + 371.37 x_{19} - 188.38 x_{20} = 34.47 \\
626.79 x_1 - 632.85 x_2 + 832.5 x_3 + 919.47 x_4 - 147.94 x_5 + 129.16 x_6 - 413.52 x_7 - 903.96 x_8 - 984.56 x_9 - 28.04 x_{10} + 807.26 x_{11} - 314.4 x_{12} + 297.48 x_{13} - 562.44 x_{14} - 238.82 x_{15} + 926.47 x_{16} - 983.03 x_{17} - 613.3 x_{18} - 971.89 x_{19} - 322.12 x_{20} = 903.16 \\
379.34 x_1 + 443.73 x_2 - 757.34 x_3 + 537.64 x_4 - 531.19 x_5 + 709.44 x_6 + 353.84 x_7 + 119.51 x_8 + 271.98 x_9 - 410.84 x_{10} + 353.19 x_{11} + 500.78 x_{12} - 556.59 x_{13} - 192.94 x_{14} - 48.36 x_{15} + 174.16 x_{16} + 784.86 x_{17} + 740.45 x_{18} + 767.39 x_{19} - 351.58 x_{20} = -836.4 \\
469.02 x_1 + 668.92 x_2 - 11.59 x_3 + 753.51 x_4 + 707.9 x_5 + 840.63 x_6 + 721.59 x_7 - 481.73 x_8 + 49.17 x_9 + 251.69 x_{10} - 571.51 x_{11} - 828.33 x_{12} - 472.26 x_{13} - 172.72 x_{14} + 2.37 x_{15} - 218.71 x_{16} + 266.92 x_{17} - 953.73 x_{18} - 97.48 x_{19} - 67.55 x_{20} = 317.83 \\
169.26 x_1 + 798.32 x_2 - 135.73 x_3 - 458.94 x_4 + 40.17 x_5 - 594.19 x_6 - 334.26 x_7 - 489.39 x_8 + 808.86 x_9 - 78.5 x_{10} + 320.69 x_{11} - 98.82 x_{12} - 576.33 x_{13} + 157.87 x_{14} - 772.42 x_{15} + 552.54 x_{16} + 304.47 x_{17} - 788.61 x_{18} - 182.78 x_{19} - 826.79 x_{20} = -402. \\
-552.42 x_1 - 113.19 x_2 - 580.43 x_3 + 990.18 x_4 - 615.94 x_5 + 718.8 x_6 - 648.02 x_7 - 826.76 x_8 + 249. x_9 - 630.9 x_{10} + 631.84 x_{11} + 649.54 x_{12} + 133.88 x_{13} + 199.25 x_{14} + 950.14 x_{15} - 377.78 x_{16} - 270.66 x_{17} - 775.68 x_{18} + 23.61 x_{19} + 306.48 x_{20} = 790. \\
-543.17 x_1 + 117.75 x_2 + 359.13 x_3 + 883.1 x_4 + 865.15 x_5 + 678.63 x_6 - 238.36 x_7 - 34.73 x_8 + 52.67 x_9 + 68.47 x_{10} + 203.12 x_{11} - 59.59 x_{12} - 492.75 x_{13} + 860.32 x_{14} - 838.9 x_{15} + 628.1 x_{16} - 9.23 x_{17} - 958.24 x_{18} + 850.51 x_{19} - 62.33 x_{20} = -745.84 \\
811.41 x_1 + 12.92 x_2 - 866.6 x_3 + 19.57 x_4 - 363.81 x_5 - 875.38 x_6 - 971.07 x_7 + 845.35 x_8 - 992.7 x_9 + 175.22 x_{10} - 860.87 x_{11} + 966.07 x_{12} - 352.11 x_{13} + 183.6 x_{14} - 933.21 x_{15} + 370.88 x_{16} - 24.71 x_{17} + 380.07 x_{18} - 670.33 x_{19} + 169.47 x_{20} = -903.4 \\
-600.08 x_1 - 512.71 x_2 - 462.16 x_3 + 167.76 x_4 + 630.07 x_5 + 523.52 x_6 - 335.63 x_7 + 15.39 x_8 - 936.2 x_9 - 356.16 x_{10} + 123.8 x_{11} - 470.64 x_{12} + 80.95 x_{13} - 387.57 x_{14} - 777.32 x_{15} - 71.62 x_{16} - 403.13 x_{17} + 519.81 x_{18} + 516.76 x_{19} + 186.9 x_{20} = -367.55 \\
378.95 x_1 - 467.7 x_2 - 861.62 x_3 - 111.97 x_4 + 220.49 x_5 + 170.97 x_6 - 533.27 x_7 - 658.91 x_8 - 979.97 x_9 + 427.84 x_{10} - 807.15 x_{11} - 849.86 x_{12} - 571.91 x_{13} - 99.13 x_{14} - 414.3 x_{15} + 645.45 x_{16} - 543.21 x_{17} - 35.32 x_{18} - 97.47 x_{19} - 324.59 x_{20} = -564.15 \\
-492.87 x_1 + 227.51 x_2 - 667.17 x_3 + 246.83 x_4 + 557.13 x_5 - 856.95 x_6 + 56.34 x_7 + 798.44 x_8 + 507.54 x_9 + 592.74 x_{10} - 110.29 x_{11} + 362.36 x_{12} - 128.01 x_{13} - 377.34 x_{14} + 638.49 x_{15} - 873.77 x_{16} - 928.8 x_{17} - 735.66 x_{18} + 882.01 x_{19} - 338.56 x_{20} = 278.43 \\
631.87 x_1 - 540.45 x_2 - 238.7 x_3 + 985.6 x_4 + 152.46 x_5 - 638.94 x_6 - 398.08 x_7 + 927.71 x_8 - 136.97 x_9 - 619.1 x_{10} - 712.02 x_{11} - 454.88 x_{12} + 328.99 x_{13} + 839.72 x_{14} - 51.3 x_{15} + 648.65 x_{16} - 4.9 x_{17} + 629.45 x_{18} - 539.45 x_{19} - 666.51 x_{20} = 483. \\
-284.08 x_1 - 965.25 x_2 + 692.73 x_3 + 435.32 x_4 - 421.15 x_5 - 477.05 x_6 - 778.96 x_7 + 858.4 x_8 - 583.6 x_9 + 639.09 x_{10} - 848.11 x_{11} - 724.28 x_{12} + 494.36 x_{13} - 698.93 x_{14} - 247.34 x_{15} + 286.78 x_{16} + 177.74 x_{17} - 158.99 x_{18} + 854.85 x_{19} - 643.42 x_{20} = -483.35 \\
-696.29 x_1 - 353.8 x_2 - 114.64 x_3 + 531.11 x_4 - 853.14 x_5 - 501.3 x_6 - 0.28 x_7 + 92.94 x_8 + 797.43 x_9 - 989.77 x_{10} - 465.1 x_{11} - 135.54 x_{12} - 210.06 x_{13} - 54.58 x_{14} - 515.13 x_{15} + 943.77 x_{16} + 199.82 x_{17} - 119.55 x_{18} - 667.3 x_{19} + 399.16 x_{20} = -197.66 \\
562.68 x_1 - 843.81 x_2 + 388.52 x_3 - 996.41 x_4 - 96.69 x_5 + 835.98 x_6 + 774.89 x_7 + 838.07 x_8 + 827.36 x_9 - 793.07 x_{10} + 84.5 x_{11} + 223.55 x_{12} + 821.56 x_{13} + 744.73 x_{14} - 94.48 x_{15} - 815.9 x_{16} + 188.7 x_{17} - 932.63 x_{18} + 174.47 x_{19} - 969.09 x_{20} = 19.54 \\
204.75 x_1 + 410.75 x_2 + 695.21 x_3 + 136.15 x_4 + 159.5 x_5 + 354.08 x_6 + 113.38 x_7 + 518.57 x_8 + 583.42 x_9 - 434.22 x_{10} + 171.11 x_{11} + 30.25 x_{12} - 879.34 x_{13} + 826.11 x_{14} + 762.62 x_{15} + 611.79 x_{16} - 947.76 x_{17} + 269.34 x_{18} - 266.05 x_{19} - 843.78 x_{20} = 808.97 \\
-686.75 x_1 + 800.09 x_2 + 546.7 x_3 - 70.72 x_4 + 553.74 x_5 - 643.98 x_6 - 111.55 x_7 + 296.34 x_8 - 864.56 x_9 + 362.37 x_{10} - 271.52 x_{11} - 604.26 x_{12} - 577.72 x_{13} - 281.44 x_{14} - 421.57 x_{15} - 318.52 x_{16} + 708.66 x_{17} - 802.64 x_{18} + 139.21 x_{19} + 180.08 x_{20} = -335.67 \\
-737.56 x_1 - 62.46 x_2 + 917.99 x_3 + 840.56 x_4 + 819.04 x_5 + 454.84 x_6 - 819.73 x_7 + 934.08 x_8 - 535.84 x_9 - 878.05 x_{10} + 606.05 x_{11} + 815.06 x_{12} - 34.63 x_{13} + 234.29 x_{14} + 887.2 x_{15} + 322.67 x_{16} - 587.67 x_{17} + 964.14 x_{18} + 348.76 x_{19} - 918.05 x_{20} = -467.79 \\
927.55 x_1 + 77.85 x_2 - 959.3 x_3 + 941.95 x_4 + 947.41 x_5 + 895.7 x_6 - 313.83 x_7 - 597.41 x_8 + 349.24 x_9 - 425.99 x_{10} - 814.86 x_{11} - 98.2 x_{12} - 671.63 x_{13} + 856.94 x_{14} + 749.07 x_{15} - 191.32 x_{16} - 979.13 x_{17} + 923.73 x_{18} - 428.21 x_{19} - 862.77 x_{20} = -232.68 \\
-525.57 x_1 - 614.14 x_2 - 906.27 x_3 + 130.58 x_4 + 387.06 x_5 - 402.56 x_6 - 72.54 x_7 - 421.9 x_8 + 958.16 x_9 - 670.52 x_{10} - 641.16 x_{11} + 614.21 x_{12} - 669.64 x_{13} - 623.88 x_{14} - 819.64 x_{15} + 816.26 x_{16} + 905.78 x_{17} + 753.09 x_{18} - 607.54 x_{19} + 720.61 x_{20} = 786.12
\end{bmatrix}$$

Before solving a linear system, it is important to analyze the properties of the system that may suggest what method is best for solving a system efficiently or if the system has a solution at all. These preliminary computations are especially important for a computer to perform because the results will indicate to the computer what is the most optimal algorithm that solves the system. For example, if a computer knows it is given a sparse linear system, a system with a coefficient matrix that has more zero than non-zero values, then the computer can allocate memory that will only store the nonzero values and perform mathematical operations on those non-zero values alone. Lack of consideration of the properties of the system may result in an inappropriate use of an algorithm that yields an answer with low precision due to round-off or truncation errors, or a solution that is generated after a long period of time.

The first step is to represent the linear system as a matrix equation $A\mathbf{x}=\mathbf{b}$ where A is an element of a 20 by 20 dimensional real number set. The most important properties of the system can be observed through matrix A especially because it makes up the bulk of the values in the system. Thus most of the analysis will be on this coefficient matrix besides the system as a whole.

$$A = \begin{pmatrix}
65.94 & -440.77 & -594.7 & 699.79 & -417.02 & 432.28 & -572.34 & -907.73 & -267.13 & -182.45 & -266.27 & 462.43 & 842.9 & 233.29 & 995.66 & 704.14 & 148.03 & 970.26 & 371.37 & -188.38 \\
626.79 & 632.85 & 832.5 & 919.47 & -147.94 & 129.16 & -413.52 & -903.96 & -984.56 & -28.04 & 807.26 & -314.4 & 297.48 & -562.44 & -238.82 & 926.47 & -983.03 & -613.3 & -971.89 & -322.12 \\
379.34 & 443.73 & -757.34 & 537.64 & -531.19 & 709.44 & 353.84 & 119.51 & 271.98 & -410.84 & 353.19 & 500.78 & -556.59 & -192.94 & -48.36 & 174.16 & 784.86 & 740.45 & 767.39 & -351.58 \\
469.02 & 668.92 & -11.59 & 753.51 & 707.9 & 840.63 & 721.59 & -481.73 & 49.17 & 251.69 & -571.51 & -828.33 & -472.26 & -172.72 & 2.37 & -218.71 & 266.92 & -953.73 & -97.48 & -67.55 \\
169.26 & 798.32 & -135.73 & -458.94 & 40.17 & -594.19 & -334.26 & -489.39 & 808.86 & -78.5 & 320.69 & -98.82 & -576.33 & 157.87 & -772.42 & 552.54 & 304.47 & -788.61 & -182.78 & -826.79 \\
-552.42 & -113.19 & -580.43 & 990.18 & -615.94 & 718.8 & -648.02 & -826.76 & 249. & -630.9 & 631.84 & 649.54 & 133.88 & 199.25 & 950.14 & -377.78 & -270.66 & -775.68 & 23.61 & 306.48 \\
-543.17 & 117.75 & 359.13 & 883.1 & 865.15 & 678.63 & -238.36 & -34.73 & 52.67 & 68.47 & 203.12 & -59.59 & -492.75 & 860.32 & -838.9 & 628.1 & -9.23 & -958.24 & 850.51 & -62.33 \\
811.41 & 12.92 & -866.6 & 19.57 & -363.81 & -875.38 & -971.07 & 845.35 & -992.7 & 175.22 & -860.87 & 966.07 & -352.11 & 183.6 & -933.21 & 370.88 & -24.71 & 380.07 & -670.33 & 169.47 \\
-600.08 & -512.71 & -462.16 & 167.76 & 630.07 & 523.52 & -335.63 & 15.39 & -936.2 & -356.16 & 123.8 & -470.64 & 80.95 & -387.57 & -777.32 & -71.62 & -403.13 & 519.81 & 516.76 & 186.9 \\
378.95 & -467.7 & -861.62 & -111.97 & 220.49 & 170.97 & -533.27 & -658.91 & -979.97 & 427.84 & -807.15 & -849.86 & -571.91 & -99.13 & -414.3 & 645.45 & -543.21 & -35.32 & -97.47 & -324.59 \\
-492.87 & 227.51 & -667.17 & 246.83 & 557.13 & -856.95 & 56.34 & 798.44 & 507.54 & 592.74 & -110.29 & 362.36 & -128.01 & -377.34 & 638.49 & -873.77 & -928.8 & -735.66 & 882.01 & -338.56 \\
631.87 & -540.45 & -238.7 & 985.6 & 152.46 & -638.94 & -398.08 & 927.71 & -136.97 & -619.1 & -712.02 & -454.88 & 328.99 & 839.72 & -51.3 & 648.65 & -4.9 & 629.45 & -539.45 & -666.51 \\
-284.08 & -965.25 & 692.73 & 435.32 & -421.15 & -477.05 & -778.96 & 858.4 & -583.6 & 639.09 & -848.11 & -724.28 & 494.36 & -698.93 & -247.34 & 286.78 & 177.74 & -158.99 & 854.85 & -643.42 \\
-696.29 & -353.8 & -114.64 & 531.11 & -853.14 & -501.3 & -0.28 & 92.94 & 797.43 & -989.77 & -465.1 & -135.54 & -210.06 & -54.58 & -515.13 & 943.77 & 199.82 & -119.55 & -667.3 & 399.16 \\
562.68 & -843.81 & 388.52 & -996.41 & -96.69 & 835.98 & 774.89 & 838.07 & 827.36 & -793.07 & 84.5 & 223.55 & 821.56 & 744.73 & -94.48 & -815.9 & 188.7 & -932.63 & 174.47 & -969.09 \\
204.75 & 410.75 & 695.21 & 136.15 & 159.5 & 354.08 & 113.38 & 518.57 & 583.42 & -434.22 & 171.11 & 30.25 & -879.34 & 826.11 & 762.62 & 611.79 & -947.76 & 269.34 & -266.05 & -843.78 \\
-686.75 & 800.09 & 546.7 & -70.72 & 553.74 & -643.98 & -111.55 & 296.34 & -864.56 & 362.37 & -271.52 & -604.26 & -577.72 & -281.44 & -421.57 & -318.52 & 708.66 & -802.64 & 139.21 & 180.08 \\
-737.56 & -62.46 & 917.99 & 840.56 & 819.04 & 454.84 & -819.73 & 934.08 & -535.84 & -878.05 & 606.05 & 815.06 & -34.63 & 234.29 & 887.2 & 322.67 & -587.67 & 964.14 & 348.76 & -918.05 \\
927.55 & 77.85 & -959.3 & 941.95 & 947.41 & 895.7 & -313.83 & -597.41 & 349.24 & -425.99 & -814.86 & -98.2 & -671.63 & 856.94 & 749.07 & -191.32 & -979.13 & 923.73 & -428.21 & -862.77 \\
-525.57 & -614.14 & -906.27 & 130.58 & 387.06 & -402.56 & -72.54 & -421.9 & 958.16 & -670.52 & -641.16 & 614.21 & -669.64 & -623.88 & -819.64 & 816.26 & 905.78 & 753.09 & -607.54 & 720.61
\end{pmatrix}$$

$$b = \begin{pmatrix} 34.47 \\ 903.16 \\ -836.4 \\ 317.83 \\ -402. \\ 790. \\ -745.84 \\ -903.4 \\ -367.55 \\ -564.15 \\ 278.43 \\ 483. \\ -483.35 \\ -197.66 \\ 19.54 \\ 808.97 \\ -335.67 \\ -467.79 \\ -232.68 \\ 786.12 \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \\ x10 \\ x11 \\ x12 \\ x13 \\ x14 \\ x15 \\ x16 \\ x17 \\ x18 \\ x19 \\ x20 \end{pmatrix}$$

First analysis: a hint of triviality

A **homogeneous** system has the form $A\mathbf{x}=0$ such that the right hand side of each equation is equal to zero. This type of system always has at least one solution known as the **trivial solution** which will be discussed further later in this document. The system given above, however, is in the form $A\mathbf{x}=\mathbf{b}$ which makes it a **nonhomogeneous** system. Such a system may have a solution set of zero, one, or infinitely many solutions. This fact does not help determine whether the nonhomogeneous system will have a solution or not, but in the case that it were homogeneous, it would be an important observation; thus, it is worth mentioning here. Because of some important similarities between $A\mathbf{x}=\mathbf{b}$ and $A\mathbf{x}=0$, this

topic will be further discussed after exploring methods for solving the linear system.

Second analysis: constraint counting

A linear system is **undetermined** when it has morce unknown variables than equations and **overdetermined** if it has more equations than variables. An underdetermined system has fewer equations than unknown variables, and it may have no solutions or infinitely many solutions. An overdetermined system has more equations than unknown variables, and it usually has no solution, unless some rows in the matrix can be *zero-ed* so that there are as many or less non-zero rows as there are unknown variables. The system defined above is neither over- nor under-determined because it has 20 equations with 20 unknowns. Thus under these constraints the linear system may then have one solution, infinitely many solutions, or no solutions.

Third analysis: does the determinant exist

The determinant of a matrix . Only a matrix that is square can have a determinant. If the determinant of the square matrix A is nonzero, then a solution exists, otherwise a solution does not exist.

$$\text{Determinant of A (Det[A])} = 8.14804 \times 10^{63}$$

Because the determinant is non-zero, the methods for solving $A\mathbf{x}=\mathbf{b}$ should yield a solution set containing one or more solutions.

Fourth analysis: existence of an inverse matrix A

This analysis concerns only the coefficient matrix A because *b*y the invertible matrix theorem, a matrix that is invertible is a **singular** matrix; if it is not invertible, it is a **nonsingular** matrix. For a matrix to be invertible, it must be square.

5. Methods for solving the unknown variables in a linear system

Below are various methods for finding solutions to linear systems with unknown variables. These methods will be demonstrated by solving the linear system introduced in the previous section.

I. Gaussian elimination with backward substitution and partial pivoting

Gaussian elimination with backward substitution is the simplest and most common direct method for solving a linear system. It is defined as a sequence of steps which can be programmatically implemented and executed a computer. The computer can deliver a solution in a predictable amount of time because the sequence of steps are almost identical for any linear system. If a system does not have a solution, the algorithm will halt when some inconsistencies appear after a step in the algorithm; thus it can itself be used to determine when a system does not have a solution without prior analysis of the linear system.

Before applying the row reduction algorithm, the linear system should be shown in the augmented matrix form $(A \mid \mathbf{b})$ as a reminder that the left and right hand side of the equations need to be manipulated while solving for x. The procedural steps that *reduce* the linear system will be referred to as the **forward phase** of the algorithm. After the forward phase has successfully been completed and all the equations have been *reduced*, the **backward phase** is executed. After backward phase terminates, the solution set of the system will be revealed with one, many, or no solutions.

Forward phase (row reduction)

Let $R_1, \ldots, R_n$ denote each row in the augmented matrix $(A \mid \mathbf{b})$. Each row reduction step can be summarized as follows:

$$\left(R_j - \left(a_{ji} \,/\, a_{ii}\right) R_i\right) \rightarrow R_j$$

This says that some row $j = i{+}1, i{+}2, \ldots, n$ below row i can be replaced by subtracting its values by the a factor of the values in row i above [1, pp. 366]. This operation is known as **row replacement**. To successfully perform the operation, the row i above row j should have a non-zero leading coefficient, or **pivot** element denoted by a_{ii}, in one of the columns. If there is no row above row j with a pivot element, then perform a **row interchange** operation. The row interchange operation should interchange a row i above row j with a row that has a pivot element or row j itself.

If it has not become obvious, the goal of the forward phase of the algorithm is to change the coefficients below the pivot element to zero. The pivot elements a_{ii} run along the diagonal, and the columns with the pivot element that are receiving this treatment are referred to as the **pivot columns**. This should effectively transform matrix A into an **upper triangular matrix**, or a matrix with nonzero values only along and above the diagonal.

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & \cdots & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & \cdots & a_{mn} & b_m \end{array}\right) \sim \left(\begin{array}{ccccc|c} a_{11} & a_{12} & \cdots & \cdots & a_{1n} & b_1 \\ 0 & a_{22} & \cdots & \cdots & a_{2n} & b_2 \\ 0 & 0 & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & a_{mn} & b_m \end{array}\right)$$

The forward phase is complete. The original matrix $(A \mid \mathbf{b})$ has been transformed into an equivalent matrix in **row echelon form** where the matrix is upper triangular and the diagonal elements are nonzero. It is important to remember that the new, row-reduced matrix is row equivalent to the original matrix because both sides of the equations were treated with elementary row operations.

By this point, we can identify the system is **consistent**, meaning it has a solution, if the matrix A has a pivot element in every row. Furthermore, if every column in A is a pivot column, which is the case when A is a square matrix, then there exists a unique solution. and whether or not b **spans** the columns of A [1, pp. 39].

Failure points

If the system has no solution, the algorithm would have aborted before producing an echelon matrix.

The algorithm would abort after a row reduction step if there is a row in the augmented matrix in the form [0 … 0 b] where b is a nonzero value because 0=**b** is a contradiction. In this event, the system is labeled as **inconsistent** and thus has no solution.

Below is the row echelon form of the system define in section 4.

```
(* Good Scripts - Dr. Vlasov *)
Aaug = Transpose[Join[Transpose[A], {b}]];

(* Convert decimal coefficients to rationals *)
(*refAaug=Rationalize@REF[Aaug];*)
refAaug = REF[Aaug];
refA = refAaug[[All, 1 ;; -2]];
refB = refAaug[[All, -1]];
rrefAaug = Rationalize@RowReduce[Aaug];
rrefA = rrefAaug[[All, 1 ;; -2]];
rrefB = rrefAaug[[All, -1]];
refSystem1 = matrixToSystem[refA, refB];

(* Extract the coefficients in the system *)
refAugCoeff = Normal@CoefficientArrays[refSystem1, vars];

DisplayForm[GridBox[{{"{", Style[Column[refSystem1], 6]}}]]
```

Backward phase (back substitution)

The backward substitution procedure begins when the matrix is in echelon form. The backward phase consists of solving an equation in the system for an unknown variable by replacing other variables in that equation by their known values. The procedure almost always begins at the last row of the system after the system has been reduced to row echelon form. If there is a row that contains all zeros, usually seen in the last row(s) after reducing the matrix, then the system has one or more **free variables**. Intuitively, the free variables are *free* to have any value and the system remains valid. The variables that lead the equations, known as **basic variables**, are determined to have a value based on other basic or free variables.

Failure points

If it is observed that a variable is assigned two different values, then it is said that a contradiction exists [2, pp. 36]. As a result, the system has no solution because it is not possible for a variable to simultaneously hold two different values.

After completion of backward substitution, the result would be a unique solution computed using Mathematica:

```
Size of solution set for Ax=b: 1

x = { x1 → 8.13608
      x10 → -7.73387
      x11 → -0.625227
      x12 → -1.67103
      x13 → -1.27248
      x14 → -2.35027
      x15 → 3.26399
      x16 → 1.52209
      x17 → -1.01244
      x18 → -2.10871
      x19 → 6.66244
      x2 → -0.120077
      x20 → 7.00351
      x3 → 2.53976
      x4 → -2.02836
      x5 → 2.53005
      x6 → -3.02625
      x7 → -3.50323
      x8 → -0.578323
      x9 → 0.959808 }
```

II. Solving for x with the inverse of A

Similar to how real numbers have a multiplicative inverse, a matrix can also have an inverse under certain conditions. For a matrix to be invertible it must be square. The matrix should also be row equivalent to the **identity matrix** which is the matrix containing only the value 1 along the diagonal. That is to say, for the matrix A to be invertible, it should be possible to reduce it to the identity matrix [2, pp. 111]. During the analysis it was discovered using Mathematica that the matrix A is nonsingular, or invertible. Applying the multiplicative inverse of A to both sides of the equation $A\mathbf{x}=\mathbf{b}$, it will be apparent that $\mathbf{x}$ can be found by multiplying A^{-1} by $\mathbf{b}$.

$A\mathbf{x} = \mathbf{b} \;\Leftrightarrow\; A^{-1}A\mathbf{x} = A^{-1}\mathbf{b} \;\Leftrightarrow\; I\mathbf{x} = A^{-1}\mathbf{b} \;\Leftrightarrow\; \mathbf{x} = A^{-1}\mathbf{b}$

Using Mathematica to solve for $\mathbf{x}$ by calculating $A^{-1}\mathbf{b}$, the solution should be similar if not the same as the solution obtained by Gaussian elimination with backward substitution.


```
method2Solution = Inverse[A].b;
StandardForm @ Column[{Row[{"     A^-1 b = x = ", method2Solution // MatrixForm}]}]
```

$$A^{-1}b = x = \begin{pmatrix} 8.13608 \\ -0.120077 \\ 2.53976 \\ -2.02836 \\ 2.53005 \\ -3.02625 \\ -3.50323 \\ -0.578323 \\ 0.959808 \\ -7.73387 \\ -0.625227 \\ -1.67103 \\ -1.27248 \\ -2.35027 \\ 3.26399 \\ 1.52209 \\ -1.01244 \\ -2.10871 \\ 6.66244 \\ 7.00351 \end{pmatrix}$$

Now that it has been computationally proven that a similar solution can be discovered using this method, it is important to question whether this method will always yield a feasible solution and whether it is a reasonable. To help answer this question, the calculation of the inverse matrix needs to be observed.

Finding the inverse A^{-1}

To find the inverse of A, a similar approach to gaussian elimination must be taken. Because it should be possible to reduce the matrix A to the identity matrix for A^{-1} to exist, a row reduction algorithm needs to be applied to the matrix. The Gaussian elimination row reduction algorithm presented in method 1 only reduced the system until A was in row echelon form. However, the matrix could have been further reduced to a form called **reduced row echelon form**.

A matrix is in reduced row echelon form when all the pivot elements are equal to 1, and all the elements above and below the pivot element are equal to 0. If the matrix is a square matrix, then the square matrix, if it is consistent, could have a pivot element in every column which implies it would be reduced to the invertible matrix if it is reduced to reduced row echelon form. If the square matrix is inconsistent, then it would not be reduced to the invertible matrix.

But what can be done with the identity matrix when A has been reduced to it? What does the identity matrix have to do with the inverse of A? During the process of reducing A, an identity matrix should receive the same sequence of row operations performed on A. After the identity has received the same sequence of row operations that transformed A into the inverse matrix, it will be transformed into A^{-1}. To guarantee that the identity matrix receives the same row operations as A, the identity matrix is augmented to matrix A.

$$(A \mid I) = \left(\begin{array}{cccc:cccc} a_{11} & a_{12} & \cdots & a_{1\,n} & 1 & 0 & \cdots & 0 \\ a_{21} & a_{22} & & a_{2\,n} & 0 & 1 & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & 0 \\ a_{n1} & a_{n2} & \cdots & a_{nn} & 0 & \cdots & 0 & 1 \end{array}\right)$$

The Gaussian elimination algorithm only reduces the matrix A to row echelon form, therefore another sequence of steps is required to further reduce the matrix A. Because the matrix A is already in upper triangular form, all that is required to transform it into the identity matrix is scaling the rows so that the pivot elements become 1 and replacement of the values above the diagonal with zeroes. Those steps can be symbolically summarized as follows:

$(a_{ii} \;/\; a_{ii})\, R_i \;\rightarrow\; R_i$ for each $i = 1, 2, \ldots, n$

$a_{ji} - (a_{ji})\, a_{ii}$ for each $i = 1, 2, \ldots, n$ and $j = i-1, i-2, \ldots, n-1$

After this is complete, the left side of the augmented matrix should be in the form of the identity matrix and the right side should be the inverse matrix A^{-1}.

$(A \mid I) \sim \left(I \mid A^{-1}\right)$

Note

The entire process of reducing A to the identity matrix is known as the **Gauss-Jordan method** [2, pp. 374]. The method can itself solve the system A**x**=**b**, but because it is integral to finding the inverse matrix and involves the Gaussian elimination algorithm introduced in the first method, it will not be independently covered in this article. The goal of the method is to find a solution to A**x**=**b** without backward substitution which is similar to the approach of finding a solution using the inverse of A.

Once the inverse matrix has been obtained, all that is required to solve for **x** is multiplication of the inverse by column vector **b**. The vector-matrix multiplication is simply the dot product between each row in A^{-1} and vector **b**.

III. LU Factorization

This method is the first in this article to introduce matrix factorization. Factoring a matrix is the process of decomposing a matrix into multiple matrices, or in the case of LU Decomposition, the decomposing the square matrix A into two matrices, L and U.

$A = LU$

Replacing A in the matrix equation $A\mathbf{x} = \mathbf{b}$,

$LU\mathbf{x} = \mathbf{b}$

To further simplify the resulting matrix equation, $U\mathbf{x}$ can be substituted with the with a vector **y** which would be the result of multiplying matrix U with vector **x**. As a result we end up with the following two equations:

$L\mathbf{y} = \mathbf{b}$

$U\mathbf{x} = \mathbf{y}$

The original matrix equation has been decomposed into two matrix equation. Now one system, $L\mathbf{y}$ =

b, has to be solved before **x** from the original equation, now in $U\mathbf{x} = \mathbf{y}$, can be solved.

The solution to the system using LU Decomposition is similar to the solution of the previous methods.

```
(* Reference [5] *)
luf = LinearSolve[A];
method3Solution = luf[b];
StandardForm @ Column[{Row[{"     x = ", method3Solution // MatrixForm}]}]
```

$$x = \begin{pmatrix} 8.13608 \\ -0.120077 \\ 2.53976 \\ -2.02836 \\ 2.53005 \\ -3.02625 \\ -3.50323 \\ -0.578323 \\ 0.959808 \\ -7.73387 \\ -0.625227 \\ -1.67103 \\ -1.27248 \\ -2.35027 \\ 3.26399 \\ 1.52209 \\ -1.01244 \\ -2.10871 \\ 6.66244 \\ 7.00351 \end{pmatrix}$$

IV. Cramer's method

Cramer's rule provides the solution of determined compatible systems of linear equations (with a single solution) by calculating determinants. It is a very fast method to solve systems, especially for 2x2 and 3x3 dimension systems. For larger dimensions, the determinants are much more cumbersome. Since the system being solved has a 20 by 20 dimension, it is possible to solve for only 1 variable without solving the whole system of equations.

When the system is square (same number of equations as of unknowns) and the matrix has an inverse, it is called Cramer's system.

```
cramers[A_, b_] := Module[{d = Det[A], a},
  Table[a = A; a[[All, k]] = b; Det[a]/d, {k, Length[A]}]]

method4Solution = cramers[A, b];
StandardForm @ Column[{Row[{"     x = ", method4Solution // MatrixForm}]}]
```

$$x = \begin{pmatrix} 8.13608 \\ -0.120077 \\ 2.53976 \\ -2.02836 \\ 2.53005 \\ -3.02625 \\ -3.50323 \\ -0.578323 \\ 0.959808 \\ -7.73387 \\ -0.625227 \\ -1.67103 \\ -1.27248 \\ -2.35027 \\ 3.26399 \\ 1.52209 \\ -1.01244 \\ -2.10871 \\ 6.66244 \\ 7.00351 \end{pmatrix}$$

6. Solution verification and analysis

According to the methods discussed above, a solution exists, thus **b** is a linear combination of $A\mathbf{x}$, or, **b** spans A which is to say that **b** can be written as a linear combination of the columns of A. However, before claiming that this is the solution to the system.

I. Gaussian elimination with backward substitution and partial pivoting

The following method is the simplest: substitute **x** with the values of the solution vector, multiply A by **x**, and verify that the left hand side of the linear equations equals the right hand side. The following are the results of the calculations using Mathematica:

```
{False , True , True , True , True , True , True , True , False , True , False , True , True , False , False , True , False , True , False , True }

Total equations satisfied by solution = 13
```

The result shows that some of the equations in the system are not satisfied by the solution produced by Gaussian elimination with backward substitution.

If the precision of the values in the calculation are increased, it is possible that the system will yield a solution that satisfies the system. The solution that was originally discovered had a precision of 5

decimal places. If the method is used with higher precision, the solution would be the values below.

```
Style@
 Row[{"x = ", DisplayForm[GridBox[{{"{", Column[method1solutionHighPrecision [[1]]]}}]]}]
```

```
x = {x1 → 8.13608073713245445654121083099196671030774165592579280662 9
     x2 → -0.12007743281399257909804809766834194695090354869763228061783
     x3 → 2.53976447409156557046665693448070693853753375868289326004 46
     x4 → -2.02836425683195770020515929211421695200086960625347304772 69
     x5 → 2.53005453572875412385483499787783344428169784688881136190 32
     x6 → -3.02624806498073106006231212865138830115731431857479030277 02
     x7 → -3.50323013153681085916320019485634968766143145121907169551 20
     x8 → -0.57832288136268977654575290749748947820039728272491222352 39
     x9 → 0.95980790255655894270310134013390612001256744438031525979 14
     x10 → -7.73386593862421029115695114040106042034077956615981877820 3
     x11 → -0.62522729496950701354252035434629433900054868726318978696 05
     x12 → -1.67103062197440498868573209594286181704169001367205277187 26
     x13 → -1.27248401412122978420791249126065604837496629795358946170 35
     x14 → -2.35027333815641416835077672032133143223625240435391150155 00
     x15 → 3.26398956985714090973391520863388835778061297386580984416 11
     x16 → 1.52208564065230259450376748410815941774686552428946191414 11
     x17 → -1.01244417536535185741212387374321108033980054310170684349 73
     x18 → -2.10871141216926413188739501953682947155923990836907195903 63
     x19 → 6.66244256946082887231031766090312525034544719969987764765 3
     x20 → 7.00351219740115121939147586833460852119345133769617669105 9}
```

Multiplying *A* by the solution above again using Mathematica would produce the following verification results:

```
{False , True , True , True , True , True , True , True , True , True , True , True , True , False , True , True , True , True , True }
```

```
Total equations satisfied by high precision solution = 18
```

Interestingly, the solution still does not satisfy 2 of the equations. This implies that Gaussian elimination with backward substitution and partial pivoting may be experiencing rounding errors due to insufficient computer memory to hold high precision values.

II. Solving for x with the inverse of A

To prove that the solution provided by the inverse method, matrix A will be multiplied by the solution. The result should be a vector equivalent to vector **b**.

```
SetPrecision[A, 60].{method2Solution /. Rule → (#2 &)}[[1]] == b
```

```
False
```

Mathematica returned that the product of A**x** is not equivalent to **b**.

III. LUDecomposition

In numerical analysis there is a topic about condition numbers that describe how susceptible a system

is to rounding errors [1, pp. 478]. One of the condition numbers is called the **infinity norm** and it can be obtained using Mathematica's LUDecomposition function. The result is:

```
(*Infinity norm is the third element returned by LUDecomposition *)
StandardForm@Column[{Row[{" Infinity norm = ", LUDecomposition[A][[3]]}]}]
```

```
Infinity norm = 371.637
```

This value is the maximum absolute row sum in the matrix *A*. It has been studied that if the condition number is much greater than 1, then the matrix may be considered **ill-conditioned** and that high precision calculations should be performed to avoid rounding errors [6]. A rounding error can be severe enough that an entry in the matrix will be changed to 0 which is quite significant if precision in the solution is important. This explains why the solution is not satisfying A**x**=**b**.

7. Detailed answer

The given system had the following coefficient matrix *A*

```
StandardForm@Column[{Row[{Style[MatrixForm[A], 7]}]}]
```

$$\begin{pmatrix}
65.94 & -440.77 & -594.7 & 699.79 & -417.02 & 432.28 & -572.34 & -907.73 & -267.13 & -182.45 & -266.27 & 462.43 & 842.9 & 233.29 & 995.66 & 704.14 & 148.03 & 970.26 & 371.37 \\
626.79 & 632.85 & 832.5 & 919.47 & -147.94 & 129.16 & -413.52 & -903.96 & -984.56 & -28.04 & 807.26 & -314.4 & 297.48 & -562.44 & -238.82 & 926.47 & -983.03 & -613.3 & -971.89 \\
379.34 & 443.73 & -757.34 & 537.64 & -531.19 & 709.44 & 353.84 & 119.51 & 271.98 & -410.84 & 353.19 & 500.78 & -556.59 & -192.94 & -48.36 & 174.16 & 784.86 & 740.45 & 767.39 \\
469.02 & 668.92 & -11.59 & 753.51 & 707.9 & 840.63 & 721.59 & -481.73 & 49.17 & 251.69 & -571.51 & -828.33 & -472.26 & -172.72 & 2.37 & -218.71 & 266.92 & -953.73 & -97.48 \\
169.26 & 798.32 & -135.73 & -458.94 & 40.17 & -594.19 & -334.26 & -489.39 & 808.86 & -78.5 & 320.69 & -98.82 & -576.33 & 157.87 & -772.42 & 552.54 & 304.47 & -788.61 & -182.78 \\
-552.42 & -113.19 & -580.43 & 990.18 & -615.94 & 718.8 & -648.02 & -826.76 & 249. & -630.9 & 631.84 & 649.54 & 133.88 & 199.25 & 950.14 & -377.78 & -270.66 & -775.68 & 23.61 \\
-543.17 & 117.75 & 359.13 & 883.1 & 865.15 & 678.63 & -238.36 & -34.73 & 52.67 & 68.47 & 203.12 & -59.59 & -492.75 & 860.32 & -838.9 & 628.1 & -9.23 & -958.24 & 850.51 \\
811.41 & 12.92 & -866.6 & 19.57 & -363.81 & -875.38 & -971.07 & 845.35 & -992.7 & 175.22 & -860.87 & 966.07 & -352.11 & 183.6 & -933.21 & 370.88 & -24.71 & 380.07 & -670.33 \\
-600.08 & -512.71 & -462.16 & 167.76 & 630.07 & 523.52 & -335.63 & 15.39 & -936.2 & -356.16 & 123.8 & -470.64 & 80.95 & -387.57 & -777.32 & -71.62 & -403.13 & 519.81 & 516.76 \\
378.95 & -467.7 & -861.62 & -111.97 & 220.49 & 170.97 & -533.27 & -658.91 & -979.97 & 427.84 & -807.15 & -849.86 & -571.91 & -99.13 & -414.3 & 645.45 & -543.21 & -35.32 & -97.47 \\
-492.87 & 227.51 & -667.17 & 246.83 & 557.13 & -856.95 & 56.34 & 798.44 & 507.54 & 592.74 & -110.29 & 362.36 & -128.01 & -377.34 & 638.49 & -873.77 & -928.8 & -735.66 & 882.01 \\
631.87 & -540.45 & -238.7 & 985.6 & 152.46 & -638.94 & -398.08 & 927.71 & -136.97 & -619.1 & -712.02 & -454.88 & 328.99 & 839.72 & -51.3 & 648.65 & -4.9 & 629.45 & -539.45 \\
-284.08 & -965.25 & 692.73 & 435.32 & -421.15 & -477.05 & -778.96 & 858.4 & -583.6 & 639.09 & -848.11 & -724.28 & 494.36 & -698.93 & -247.34 & 286.78 & 177.74 & -158.99 & 854.85 \\
-696.29 & -353.8 & -114.64 & 531.11 & -853.14 & -501.3 & -0.28 & 92.94 & 797.43 & -989.77 & -465.1 & -135.54 & -210.06 & -54.58 & -515.13 & 943.77 & 199.82 & -119.55 & -667.3 \\
562.68 & -843.81 & 388.52 & -996.41 & -96.69 & 835.98 & 774.89 & 838.07 & 827.36 & -793.07 & 84.5 & 223.55 & 821.56 & 744.73 & -94.48 & -815.9 & 188.7 & -932.63 & 174.47 \\
204.75 & 410.75 & 695.21 & 136.15 & 159.5 & 354.08 & 113.38 & 518.57 & 583.42 & -434.22 & 171.11 & 30.25 & -879.34 & 826.11 & 762.62 & 611.79 & -947.76 & 269.34 & -266.05 \\
-686.75 & 800.09 & 546.7 & -70.72 & 553.74 & -643.98 & -111.55 & 296.34 & -864.56 & 362.37 & -271.52 & -604.26 & -577.72 & -281.44 & -421.57 & -318.52 & 708.66 & -802.64 & 139.21 \\
-737.56 & -62.46 & 917.99 & 840.56 & 819.04 & 454.84 & -819.73 & 934.08 & -535.84 & -878.05 & 606.05 & 815.06 & -34.63 & 234.29 & 887.2 & 322.67 & -587.67 & 964.14 & 348.76 \\
927.55 & 77.85 & -959.3 & 941.95 & 947.41 & 895.7 & -313.83 & -597.41 & 349.24 & -425.99 & -814.86 & -98.2 & -671.63 & 856.94 & 749.07 & -191.32 & -979.13 & 923.73 & -428.21 \\
-525.57 & -614.14 & -906.27 & 130.58 & 387.06 & -402.56 & -72.54 & -421.9 & 958.16 & -670.52 & -641.16 & 614.21 & -669.64 & -623.88 & -819.64 & 816.26 & 905.78 & 753.09 & -607.54
\end{pmatrix}$$

Represented as vector **b**, the right hand side of the given system had the following values:

```
StandardForm @ Column[{Row[{Style["     b = ", Bold], Style[MatrixForm[b], 10]}]}]
```

$$b = \begin{pmatrix} 34.47 \\ 903.16 \\ -836.4 \\ 317.83 \\ -402. \\ 790. \\ -745.84 \\ -903.4 \\ -367.55 \\ -564.15 \\ 278.43 \\ 483. \\ -483.35 \\ -197.66 \\ 19.54 \\ 808.97 \\ -335.67 \\ -467.79 \\ -232.68 \\ 786.12 \end{pmatrix}$$

Given linear system No. 01, the linear system was found to be consistent because every method used to solve the system yielded a solution. Every method output the same solution. It was also verified at the solution, represented by vector **x** satisfied the system when it was plugged in.

The solution of the system is unique and it is:

```
Style @ Row[{"x = ", DisplayForm[GridBox[{{"{", Column[method1solution [[1]]]}}]]}]
```

$$x = \left\{ \begin{array}{l} x1 \to 8.13608 \\ x10 \to -7.73387 \\ x11 \to -0.625227 \\ x12 \to -1.67103 \\ x13 \to -1.27248 \\ x14 \to -2.35027 \\ x15 \to 3.26399 \\ x16 \to 1.52209 \\ x17 \to -1.01244 \\ x18 \to -2.10871 \\ x19 \to 6.66244 \\ x2 \to -0.120077 \\ x20 \to 7.00351 \\ x3 \to 2.53976 \\ x4 \to -2.02836 \\ x5 \to 2.53005 \\ x6 \to -3.02625 \\ x7 \to -3.50323 \\ x8 \to -0.578323 \\ x9 \to 0.959808 \end{array} \right.$$

The solution of the given system was performed in *Mathematica* using the following methods:

1. Gaussian elimination with backward substitution

2. Inverse method (multiplying **b** by of A^{-1})
3. LU Decomposition
4. Cramer's Rule

8. Conclusions

Research was done on each individual method for solving linear systems with the intention of finding pros and cons between each method.

Research findings

It was discovered that all four methods used to solve the system produced the same solution. This was surprising because some of the methods involve a lot more calculations than others. This may imply that Mathematica is using very high precision during dot product multiplication.
Discuss the broader implications of those findings.

After solving the linear system, we can conclude that the linear system is consistent because it has a solution. This became obvious transforming the augmented matrix using Gaussian elimination when the augmented matrix and no row was in the form 0=**b**.

Unique, or it has only one solution. Uniqueness can be identified by observing if every column in the coefficient matrix is a pivot column which implies there is only one solution.

When the inverse of matrix A was calculated using Mathematica, the majority of the values in the matrix were very small. Most of the entries were smaller than .001 and one was below .000001. This is concerning because if Mathematica were not using high precision values during intermediary operations, then it is possible that some of the entries in the matrix would have been changed to 0.

Research weaknesses

The report was weak on exposing the speed at which the methods can solve a linear system. It also did not go into detail about how stable the methods are because that would require knowing exactly how Mathematica implements the methods and the floating point precision used during computations. These are important topics in numerical analysis, and they should be considered by engineers writing the algorithms that solve linear systems.

Future research

It would be fun to implement methods for solving (non-)linear systems in C/C++ to further explore the memory consumption and execution times. It would be interesting to write some code that can automate the analysis of linear systems such as finding the largest element in a matrix. Open source libraries like LAPACK, Armadillo, or TNT which implement numerical methods for solving linear systems in C++ would be good libraries to learn if linear systems need to be solved on embedded systems or very large systems need to be solved.

In summary, it has been concluded that:

1. Solutions to linear systems solved computationally are not guaranteed to produce results that satisfy the original system. Though the results may be good approximations.
2. Multiple methods need to verify that a solution found by one method can be found for another. This reinforces the fact that there is (or there is not) a solution.

References

1. R. Burden, J. Faires, and A. Burden, "Numerical Analysis", 9th Ed. Cengage learning 2010, pp. 351-417.
2. D. Lay, S. Lay, and J. McDonald, Linear algebra and its applications, 6th ed. Pearson Education, 2020, pp. 17-20, 115, 113.
3. M. Ruckert, Understanding MP3, 2nd ed. Wiesbaden [Allemagne]: Vieweg, 2005.
4. "001-Good scripts (Mathematica nb-file)", Fiu.instructure.com. [Online]. Available: https://fiu.instructure.com/courses/89558/files/14136897/preview?verifier=fRJPkfEv8GtLnpySpxxQisAIjSJjw78XvYfnPaj3. [Accessed: 30- Mar- 2021].
5. Wolfram MathWorld, reference.wolfram.com [Online]. Available: https://reference.wolfram.com/language/ref/LUDecomposition.html?view=all
6. Loughborough University[Online]. Available: https://learn.lboro.ac.uk/archive/olmp/olmp_resources/pages/workbooks_1_50_jan2008/Workbook30/30_4_mtrx_norms.pdf