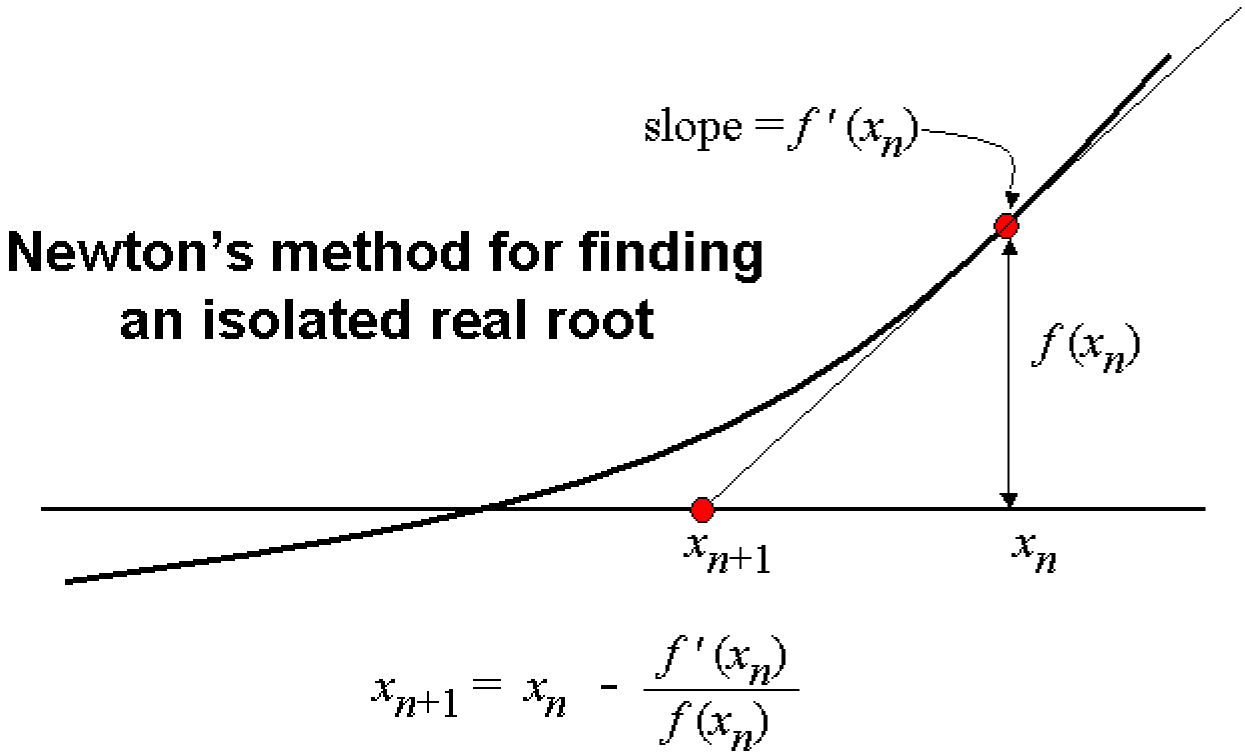
Newton Raphson

Used to solve f(x) = 0. If f(x) , f'(x) , f''(x) are continuous, then we can use this information to find the solution of
f(x)



http://www.chem.sc.edu/faculty/morgan/resources/acidbase/nrdoc.html

From the plot

m = f'(x0)

m = [f(x1) - f(x0)] / [x1 - x0]

x1 = x0 - f(x0)/f'(x0)

Newton Raphson and Taylor polynomial

We can approximate a continuous function using a Taylor polynomial expansion around x0

f(x) = f(x0) + f'(x0)(x1 - x0) + [f''(x0)(x1 - x0)2 / 2!] ...

Newton raphson approximates f(x) by using Taylor expansion up to the fires two terms

We will build the Newton raphson method again but using the Taylor expansion

f(x1)=f(x0)+f'(x0)(x1-x0),f(x1)=0

In fact, we could improve Newton-Raphson by using three terms in the Taylor expansion to construct an improved Newton raphson method that also uses the second derivative.

This method produces a function: x1=g(x0), when g(x) is quadratic not linear.

E.X.

sin (x) = 1/2, x is in radians

f(x) = sin (x) - 1/2 = 0, f'(x) = cos (x)

I	X	F(xk)	F'(xk)	X(k+1)
0	0	-0.5	1	0.5
1	0.5	-0.02	0.8795	0.522
2	0.522	-0.00139	0.8668	0.5235

Exact solution: x=0.5235

This is more precise and faster than the false position method and the bisect method

Order of Convergence

Assume that p is a solution for f(x) = 0. Also assume that E = p - pn is the approximation error. If two positive constants, A is not 0 and R > 0 exist such that

Lim n -> ? {| p – pn+1 | / | p – pn |R }

= Lim n -> ? {En+1 / (En) R }

= A

Then the sequence is said to converge to p with an order of convergence R. This means,

|En+1| = A.|En|R

R will tell us how fast a method converges

R=1: convergence is linear

R=2: convergence is quadratic

usually the more roots a question has, the slower the convergence is

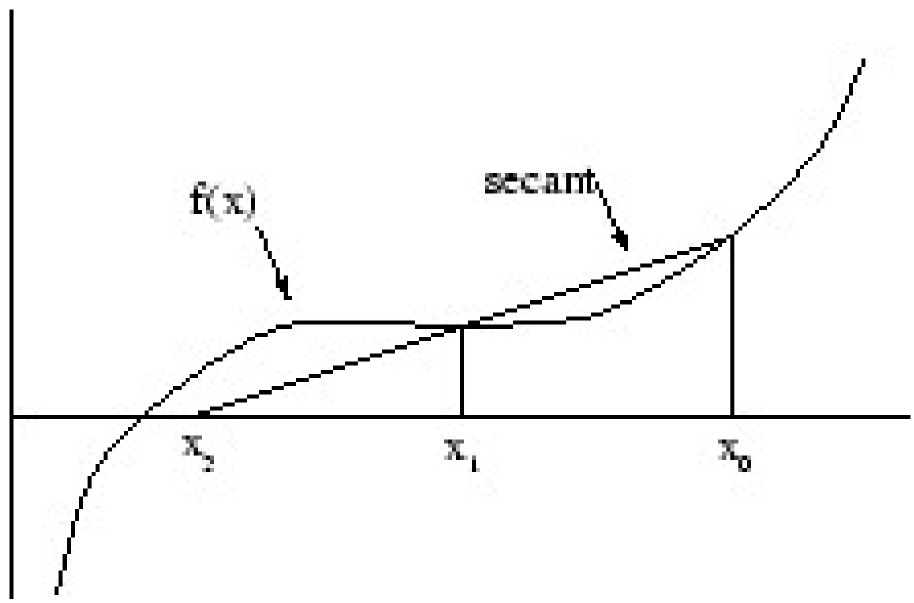
if P is a simple solution, the Newton raphson has R=2 as quadratic

Secant Method

The Newton-Raphson method requires the evaluation of two functions at each iteration and the derivative of the function

The Secant method only needs one evaluation of f(x)

if f(x) has only one root, R=1.618, which is close to Newton raphson



http://pathfinder.scar.utoronto.ca/~dyer/csca57/book_P/node36.html

m = (f(x1) - f(x0)) / (x1 – x0)

xk+1 = xk - f(xk) * (((xk – xk-1)) / (f(xk) - f(xk-1)))

I	X	F(xk)	X(k+i)
0	0	-0.5	0.03414
1	1	0.03415	0.594
2	0.594	0.597	0.50798
3	0.50798	-0.0135	0.5238

The solution of linear system

AX=B

E.X.

6x+3y+2z=29

3x+y+3z=17

2x+3y+7z=21

$$\begin{bmatrix} 6 & 3 & 2 \\ 3 & 1 & 2 \\ 2 & 3 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{matrix} 29 \\ 21 \\ 21 \end{matrix}$$

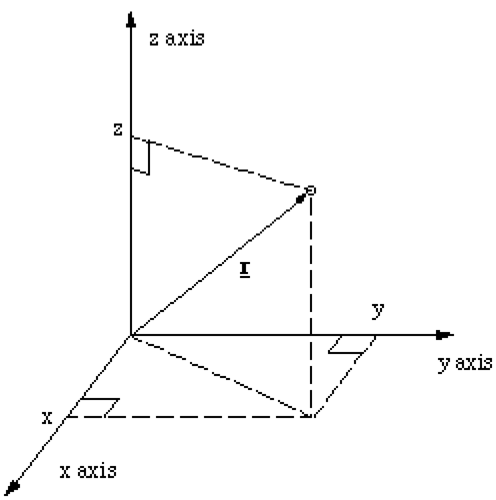
$$A \quad x = b$$

http://www.cs.purdue.edu/homes/cs314/

N Dimensional Vector: x=(x1, x2, x3......xn)

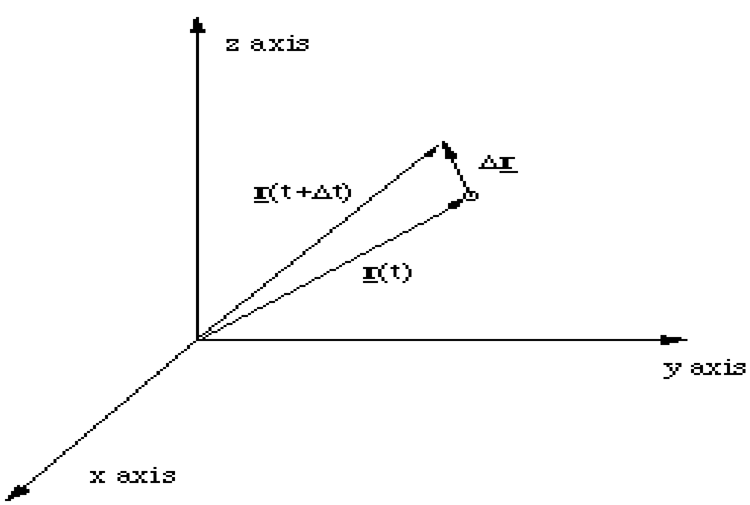
N Dimensional space: the set of all N dimensional vectors

When a vector is used to determine a position it is called a position vector



http://www.ch.ic.ac.uk/local/physical/mi_2.html

When a vector is used to denote movement between two points is space it is called the displacement vector



http://www.ch.ic.ac.uk/local/physical/mi_2.html

Properties of vectors

Equality: x=y only if xj = yj for all j = 1,2,3.....n

Sum: X + Y = x1 + y1 , x2 + y2 , x3 + y3 ... xn + yn

Negative: -Y = - y1 , - y2 , - y3 ... - yn

Difference: X - Y = x1 - y1 , x2 – y2 , x3 – y3 ... xn – yn

Scalar multiplication: cx = cx1 , cx2 , cx3... cxn

Linear combination: cx + dy = cx1 + dy1 , cx2 + dy2 , cx3 + dy3 ....cxn + dyn

Dot product: X . Y = x1.y1 , x2.y2 , x3.y3 ... xn.yn

Norm or Euclidean norm: $\mathbf{X} = (x_1^2 + x_2^2 + x_3^2 \ldots x_n^2)^{1/2}$

In scalar multiplication cx, if |c| > 1, then cx is a stretched version of X and if |c| < 1, then cx is a shrunken version of X. |CX|=X.X

Distance of point in space: X - Y = x1 - y1 , x2 – y2 , x3 – y3 ... xn – yn . $\mathbf{X} = (x_1^2 + x_2^2 + x_3^2 \ldots x_n^2)^{1/2}$

E.X.

x= (-1,3,2), y= (3,5,2)

x+y =(2,8,4)

x-y=(-4,-2,0)

2x=(-2,6,4)

|x|=(-2,6,4)

Displacement: y-x=(4,2,0)

Distance: |y-x|=20^.5

Sometimes we writ vectors in columns

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

http://wpcontent.answers.com/math/3/8/1/381890b4e3dc92930bff4ab8da4e5e5e.pngb

We use superscript “$^{\prime\prime}$” or “^{T}” to convert a vector form a row to column, vice versa.

Vector algebra

Commutative: $Y + X = X + Y$

Additive identity: $X = X + 0 = 0 + X$

Additive inverse: $X - X = X + (-X) = 0$

associative: $X + (Y + Z) = (X + Y) + Z$

Distributive with scalar: $(a + b) X = aX + bX$

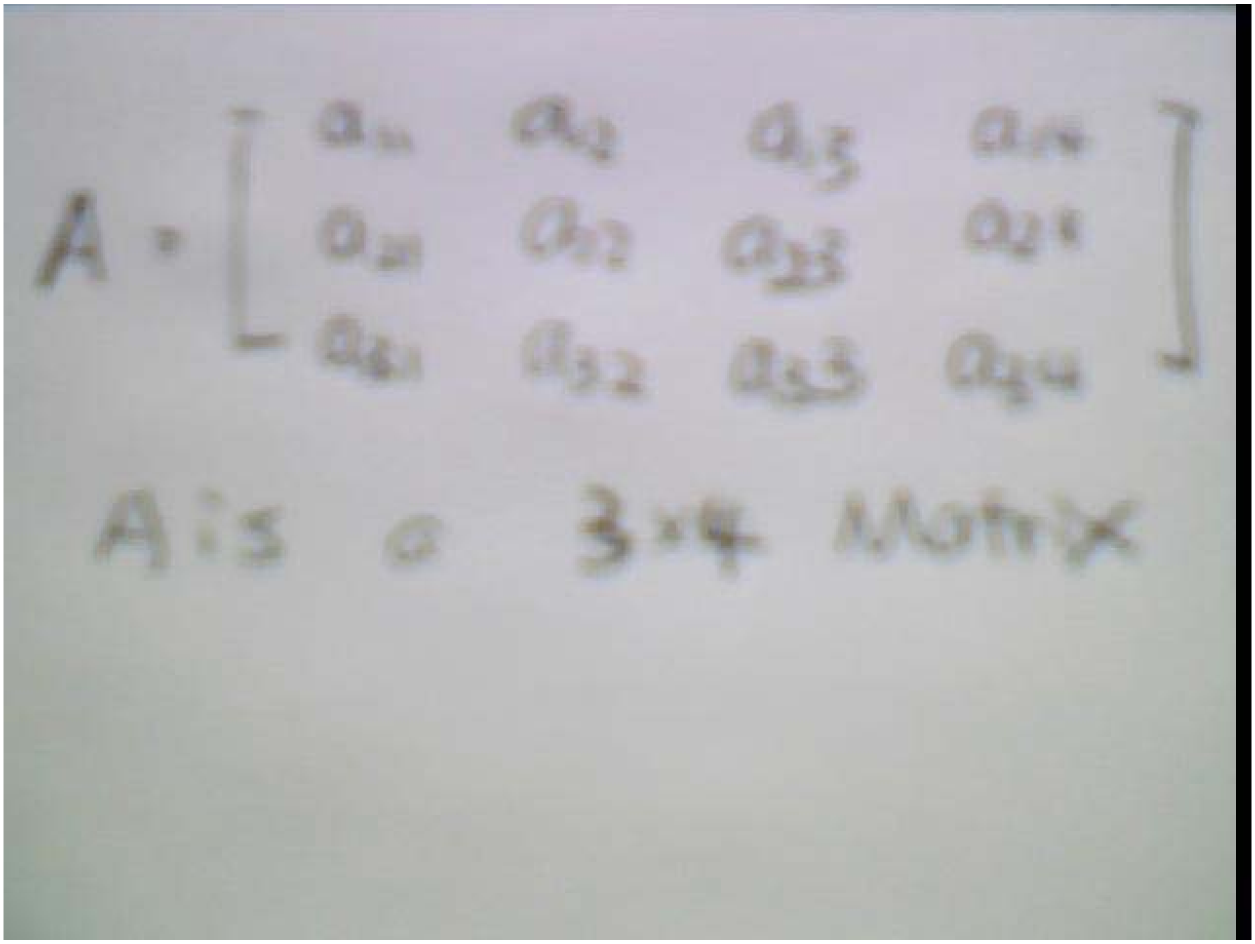
Distributive with vector: $a(X + Y) = aX + aY$

Associative with vector: $a(bX) = (ab)X$

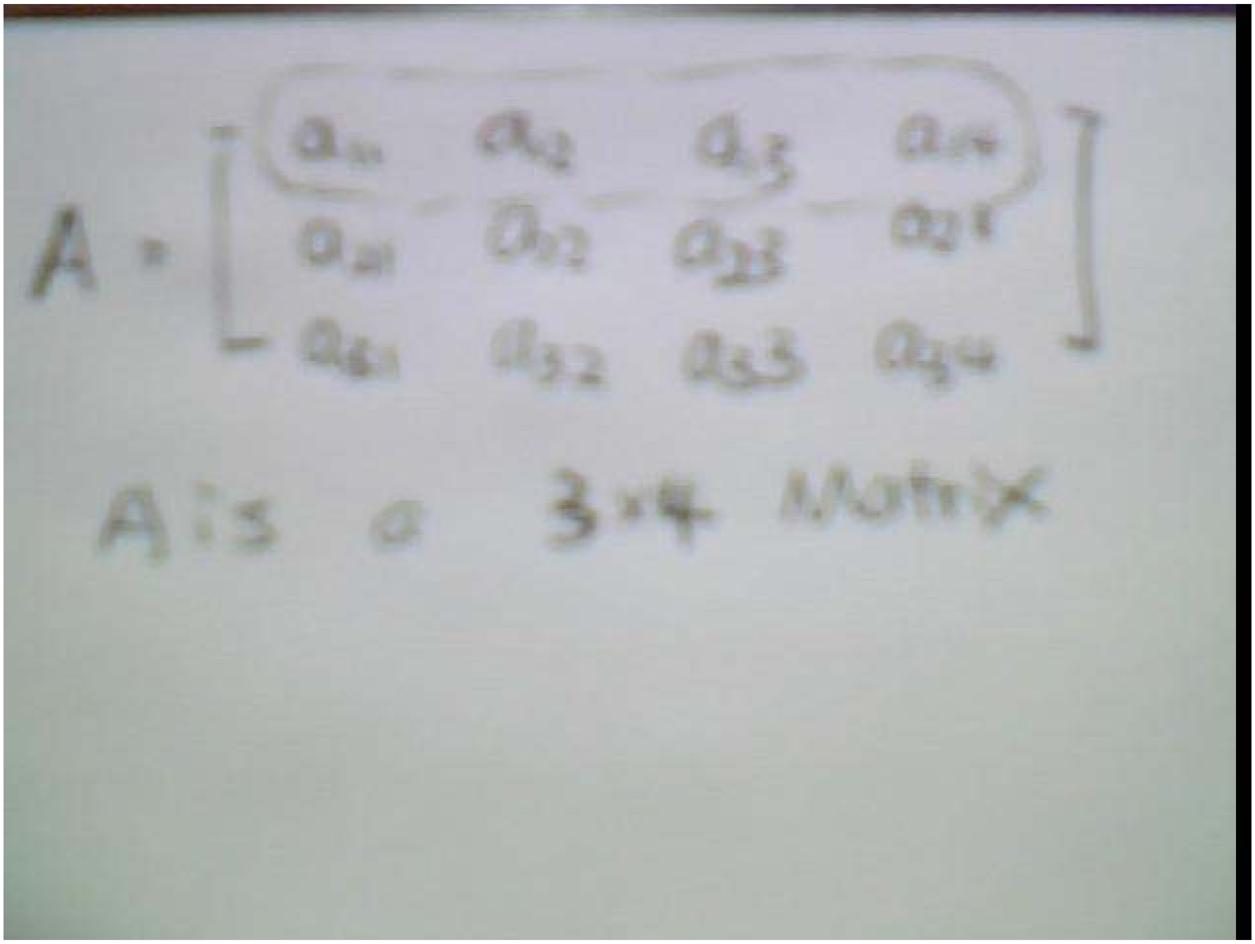
Matrices

If A is a matrix, then a_{ij} denote the number at I, and j(I is the row and j is the column)

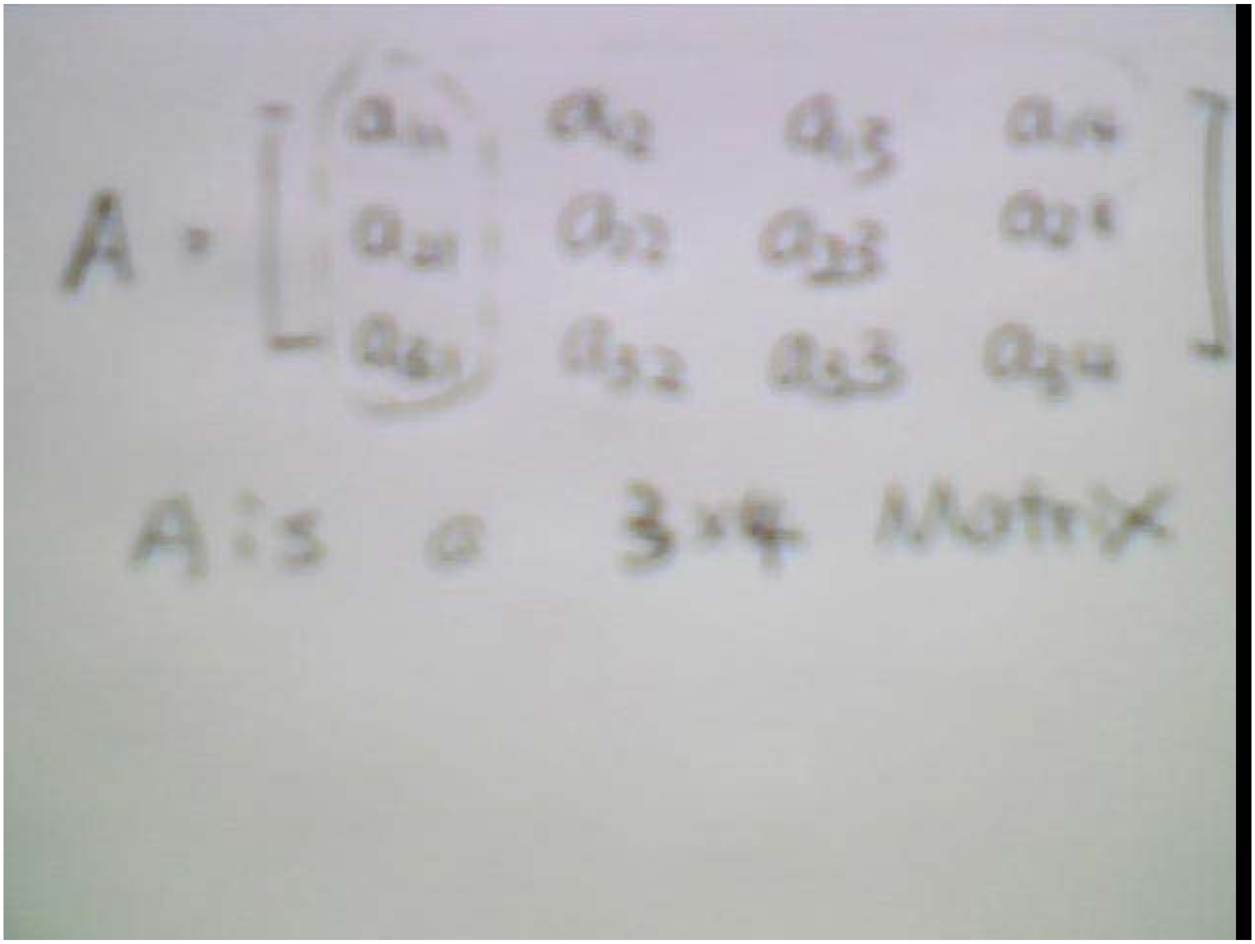
If A is a M x N matrix, then it has M rows and N columns



You can see a M*N matrix as M rows of N dimensional vector



Or as N columns of M dimensional vector



Operations with matrices

Equality: A = B only if aij = bij

Sum: A + B = $(aij + bij)_{mxn}$

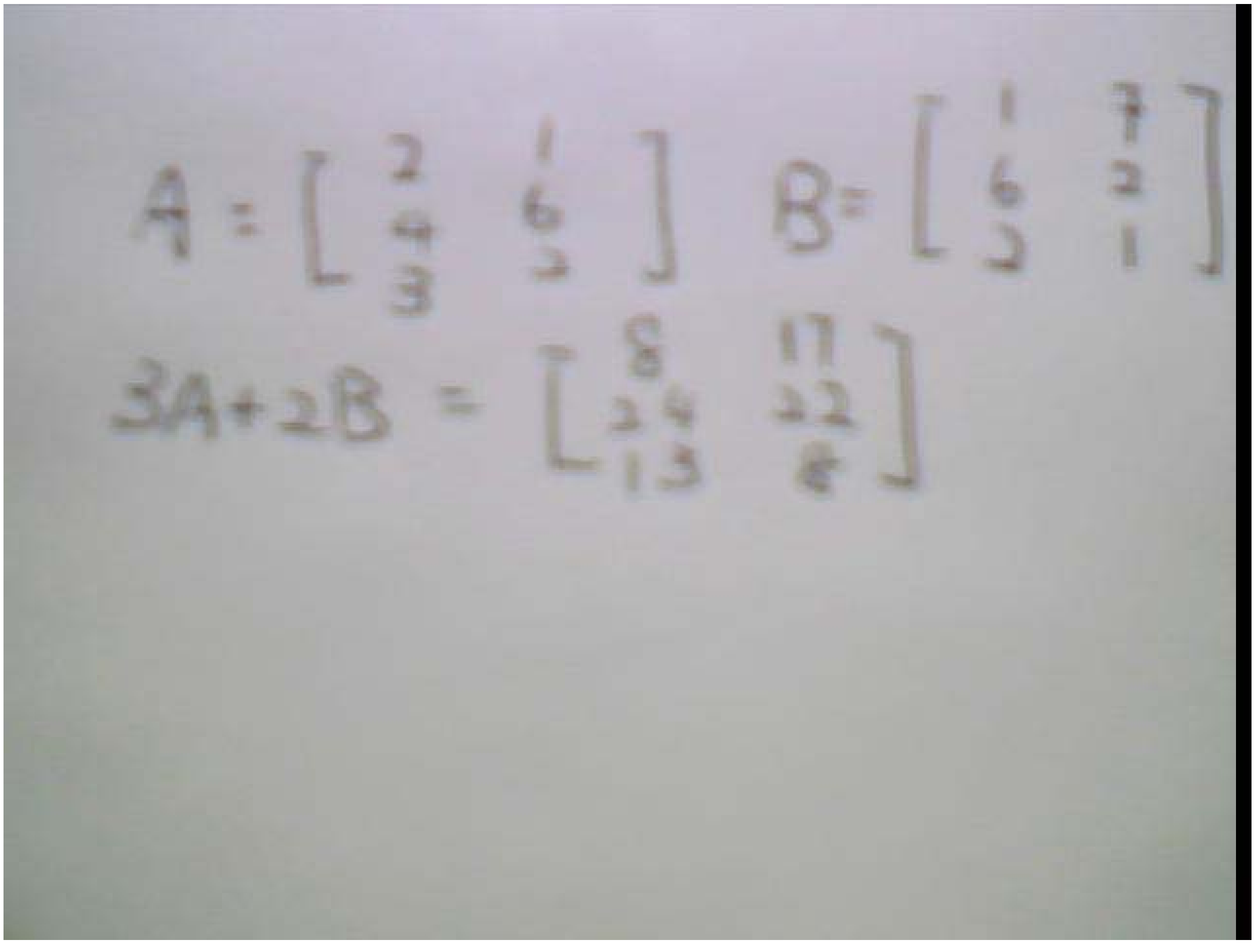
Negation: -A = $(-aij)_{mxn}$

Difference: A − B = $(aij − bij)_{mxn}$

Scalar multiplication: cA = $(caij)_{mxn}$

Weighted Sum: pA + qB = $(paij + qbij)_{mxn}$

E.X.



Properties

Commutative : X + Y = Y + X

Additive identity : X + 0 = 0 + X

Additive inverse : X − X = X + (−X) = 0

Associative : X + (Y + Z) = (X + Y) + Z

Distributive with scalars : (a + b) X = aX + bX

Distributive with matrices : a(X + Y) = aX + aY

Distributive with matrices: a(bX) = (ab)X

Matrix Multiplication

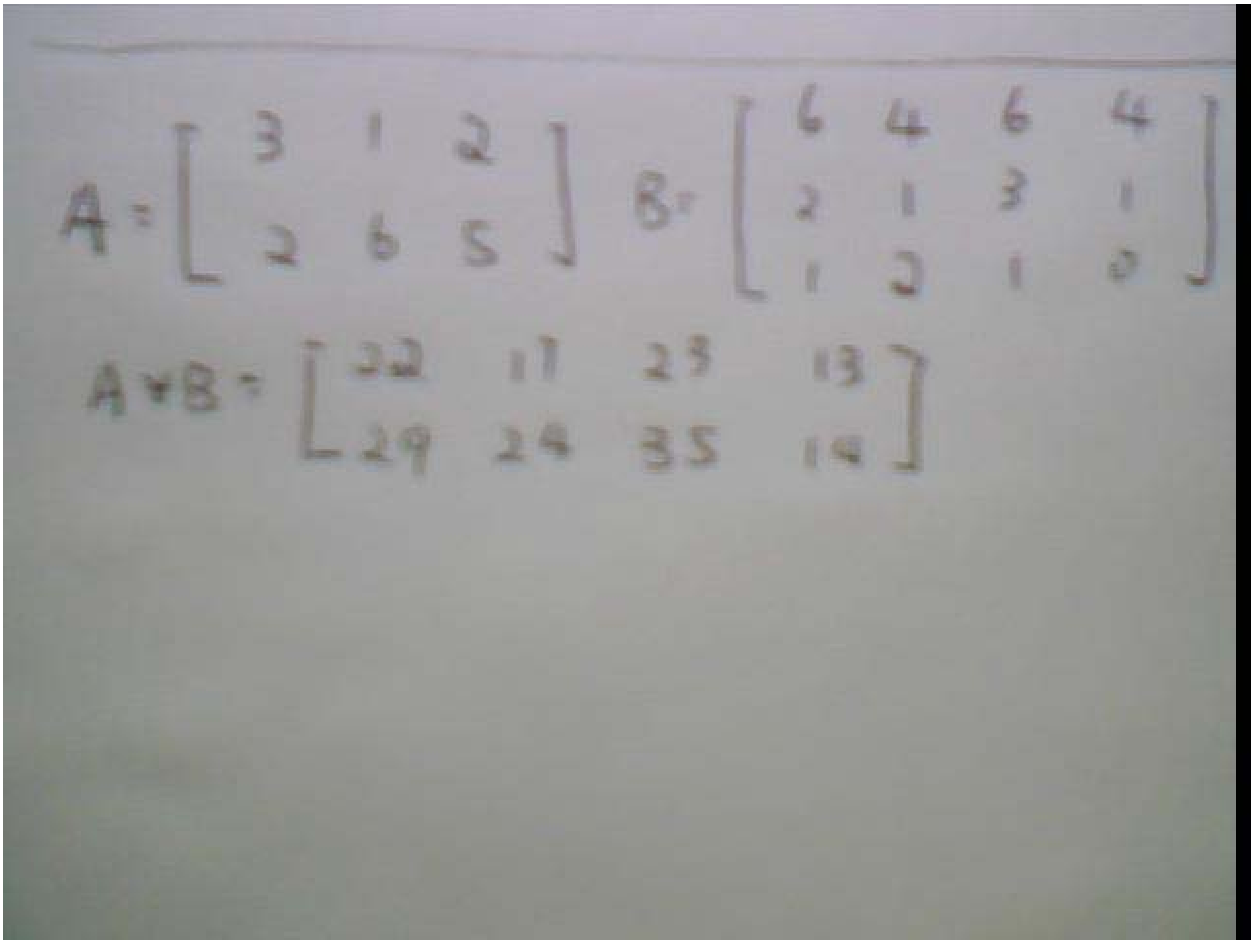
A.B=C

$A=[a_{ik}]$, M*N

$B=[_{kj}]$N*P

$C=\Sigma\ a_{jk.} b_{kxj}$, M*P

E.X.



Special matrices

Zero matrix:

$$M = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

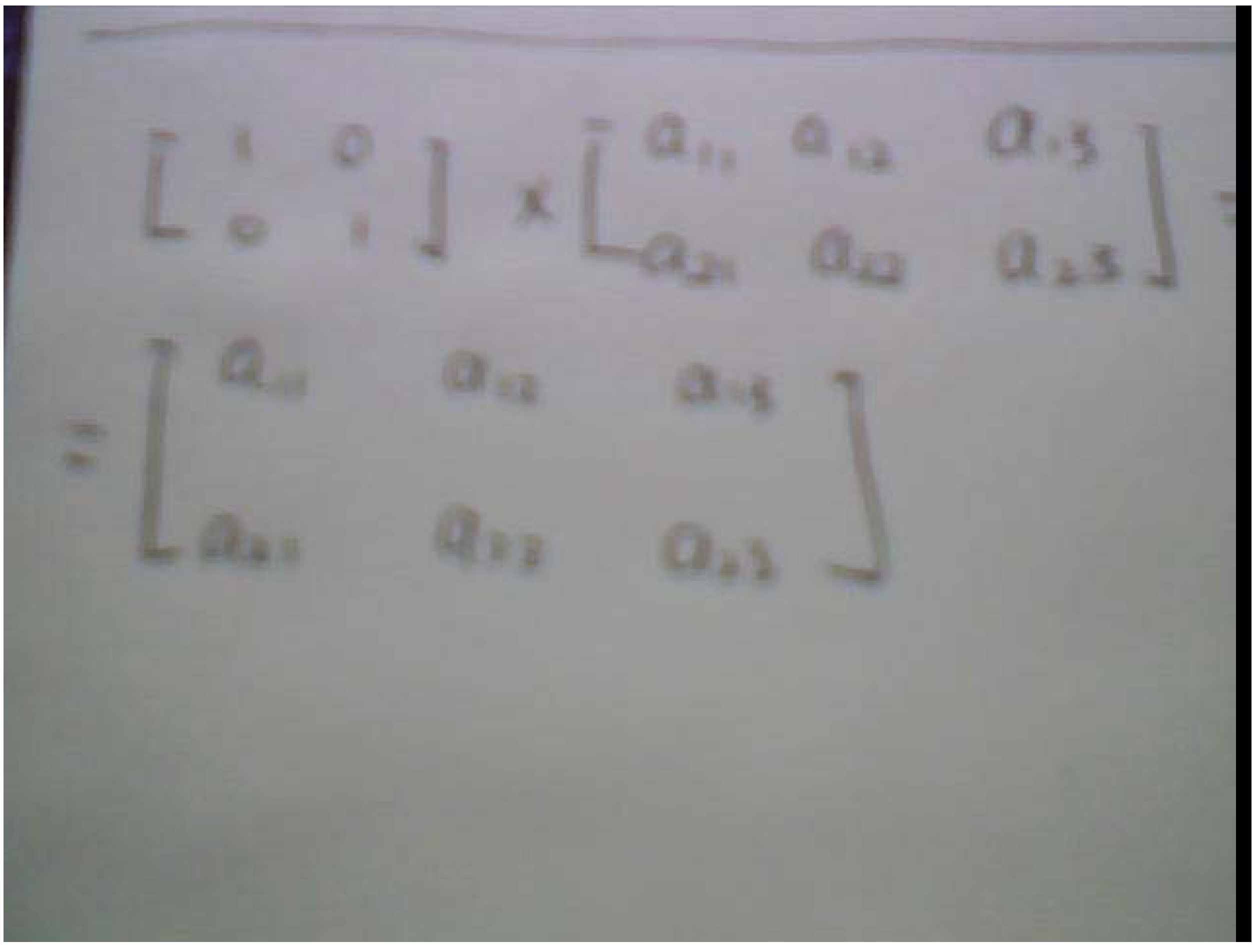
http://upload.wikimedia.org/math/6/6/3/6638ff38b6275ae82451b617f58e5b96.png

Identity matrix: $I_n = (a_{ixj})_{mxn} =$ where $(a_{ixj}) = 1$ when $i = j$ and $= 0$ when $i \neq j$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

http://help.adobe.com/en_US/FlashPlatform/beta/reference/actionscript/3/images/identityMatrix.jpg

E.X.



Properties of matrix multiplication

(AB)C = A(BC) : associative for multiplication

IA = AI = A: identity matrix

A(B + C) = AB + AC : left distributive

(A + B).C = AC + BC: right distributive

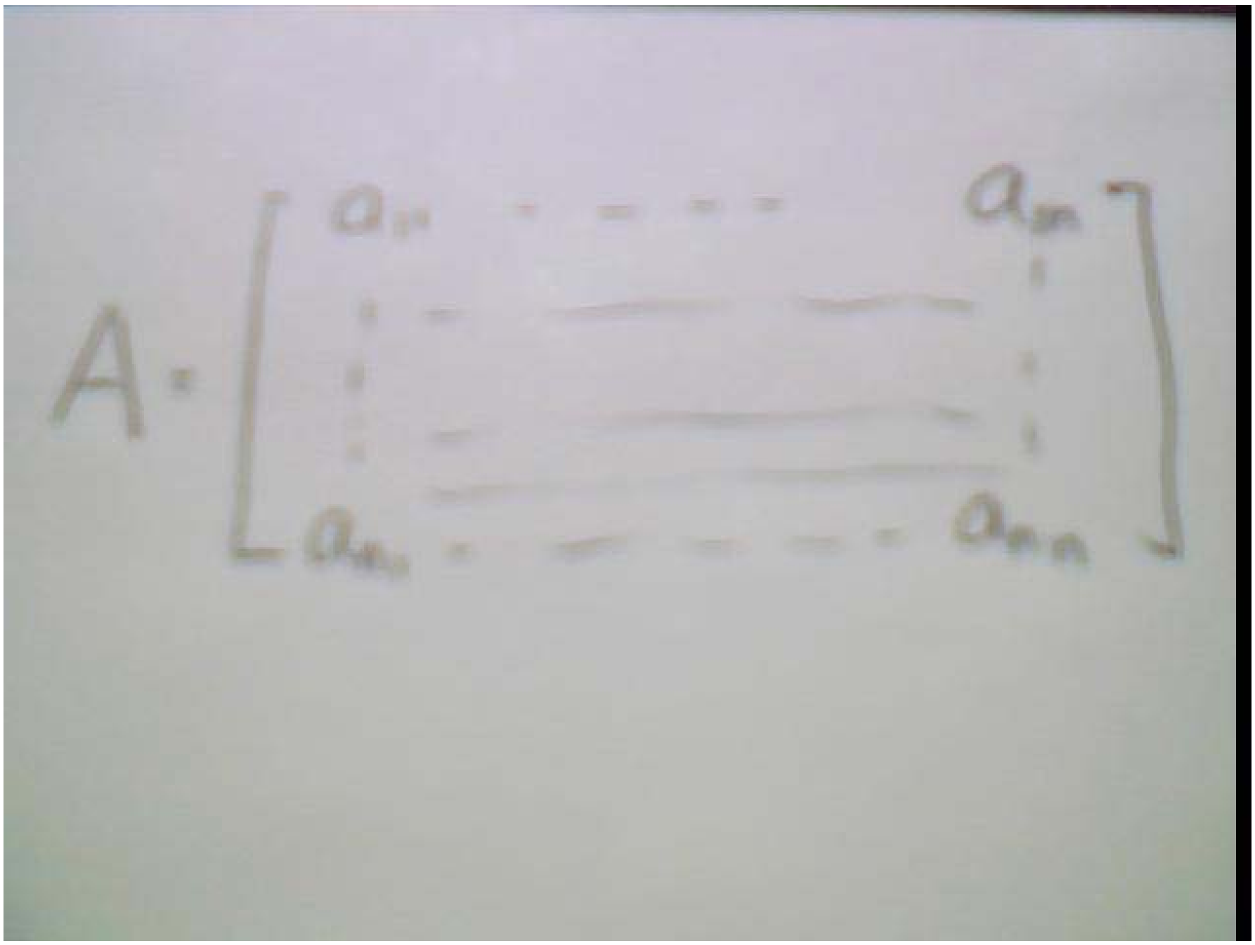
Note: No commutative property: A*B is not B*A

Inverse of a nonsingular matrix

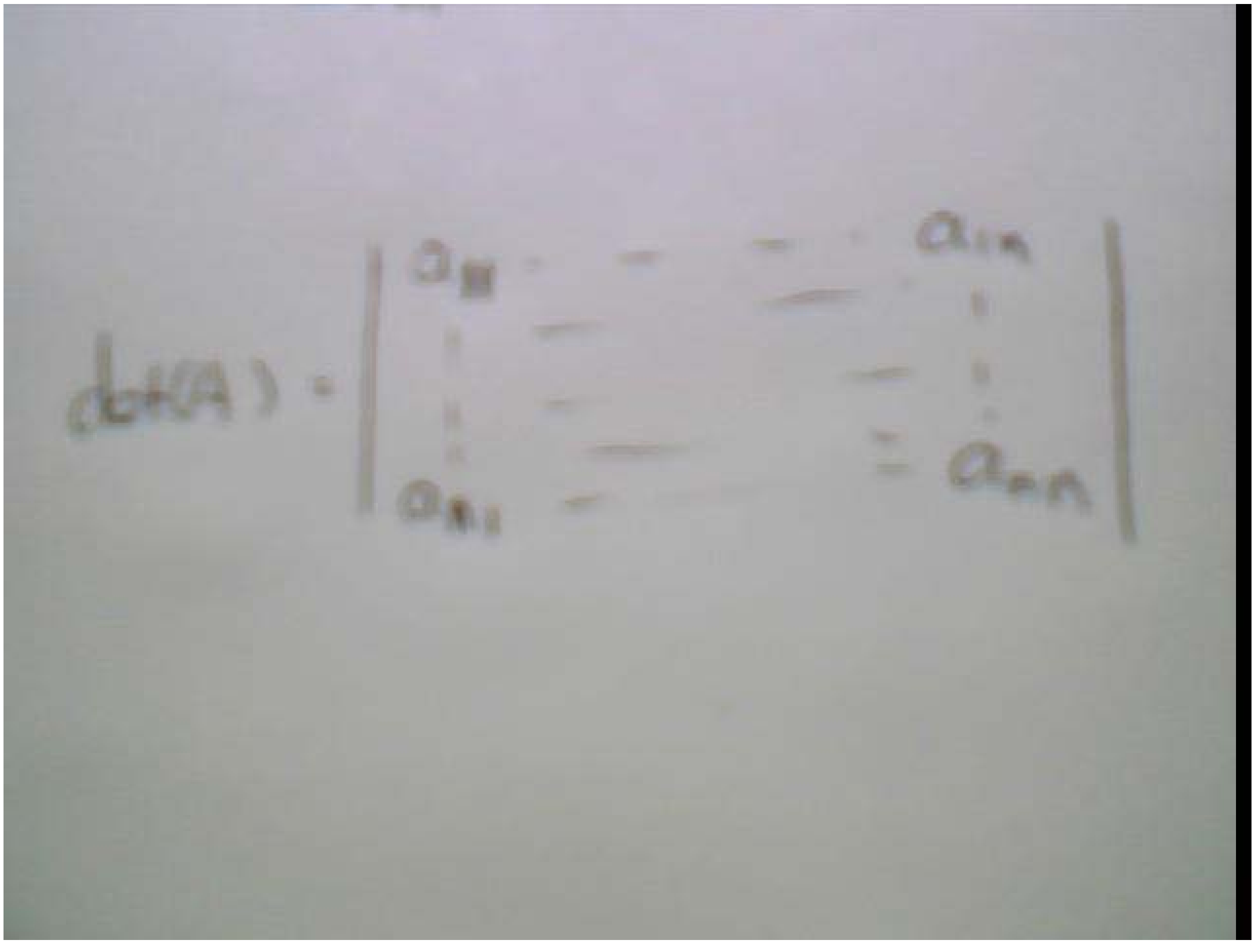
A nxn matrix A is called non-singular or invertible if there exist a N x N matrix such that A.B = B.A = I, if there is such B, A is singular, If B can be found, we can write B=A^-1, A= B^-1 or A*A^-1=A^-1*A.

Determinants

Assume



that

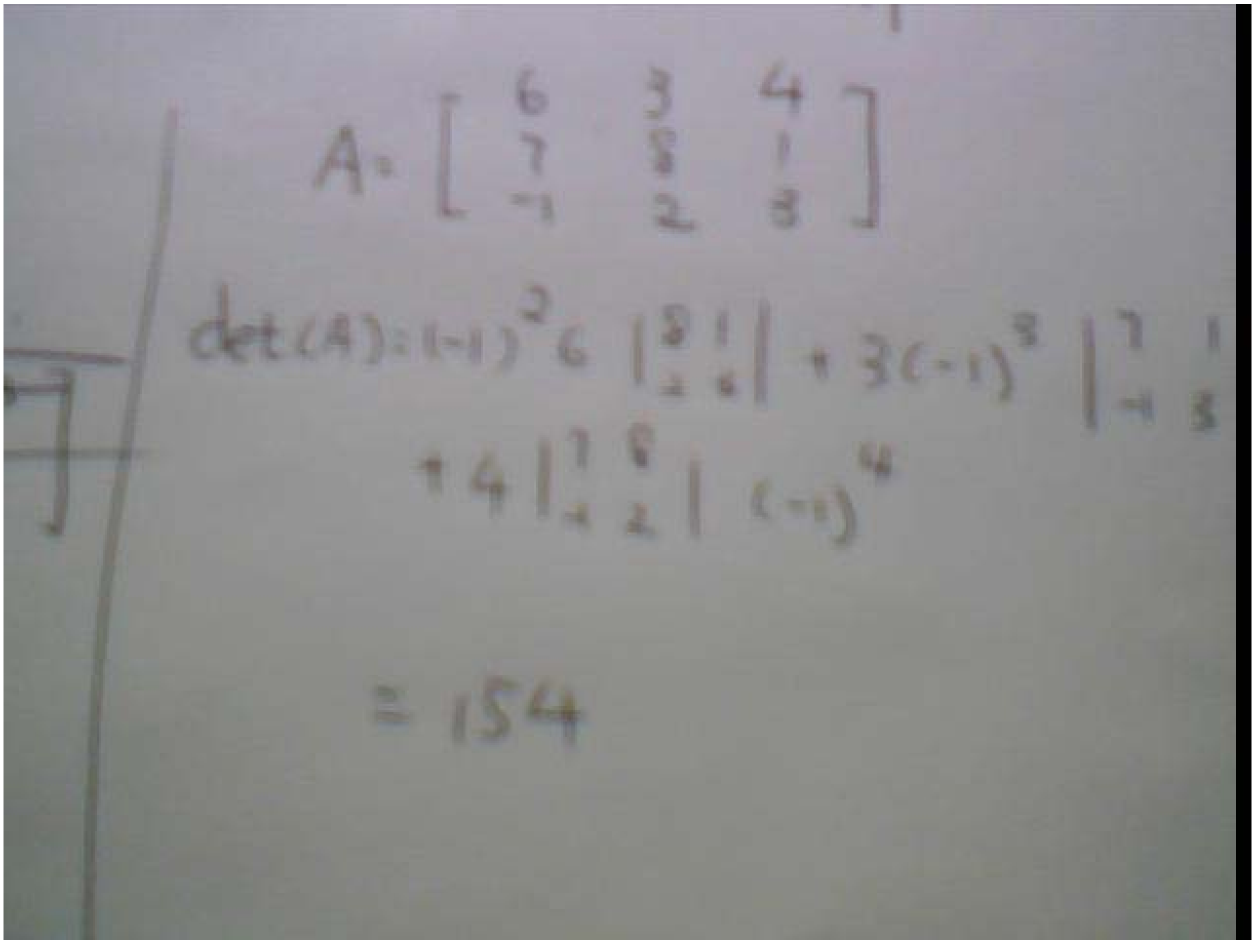


Determinants is scalar

If A= [aij], det(a)=aij

Defining determinants in a recursive way, elimiting row I and column j from A.

E.X.



Upper Triangular system

A=[aij] is called upper triangular if aij=0 when i>j.

$$U_5 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 & 6 \\ 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix};$$

E.X.

Lower triangular

A=[aij] is called lower triangular if aij=o when i<j

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{bmatrix}$$

E.X.

Upper triangular system of linear equations

$$\begin{aligned}
a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \cdots + a_{1,n-1}x_{n-1} + a_{1,n}x_n &= b_1 \\
a_{2,2}x_2 + a_{2,3}x_3 + \cdots + a_{2,n-1}x_{n-1} + a_{2,n}x_n &= b_2 \\
a_{3,3}x_3 + \cdots + a_{3,n-1}x_{n-1} + a_{3,n}x_n &= b_3 \\
\vdots \qquad\qquad \vdots \quad &\quad \vdots \\
a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n &= b_{n-1} \\
a_{n,n}x_n &= b_n
\end{aligned}$$

Backward substitution: solve for Xn in the last equation, plug it into the second last equation

Xk=(bkj=k+1najk*Xj)/akk

E.X. 3x+4y+z=2

3y+2z=1

5z=10

Using backward substitution, we get z=2, y=-1, x=4/3.

Guass Elimination

Usually system of linear equations are not upper triangular

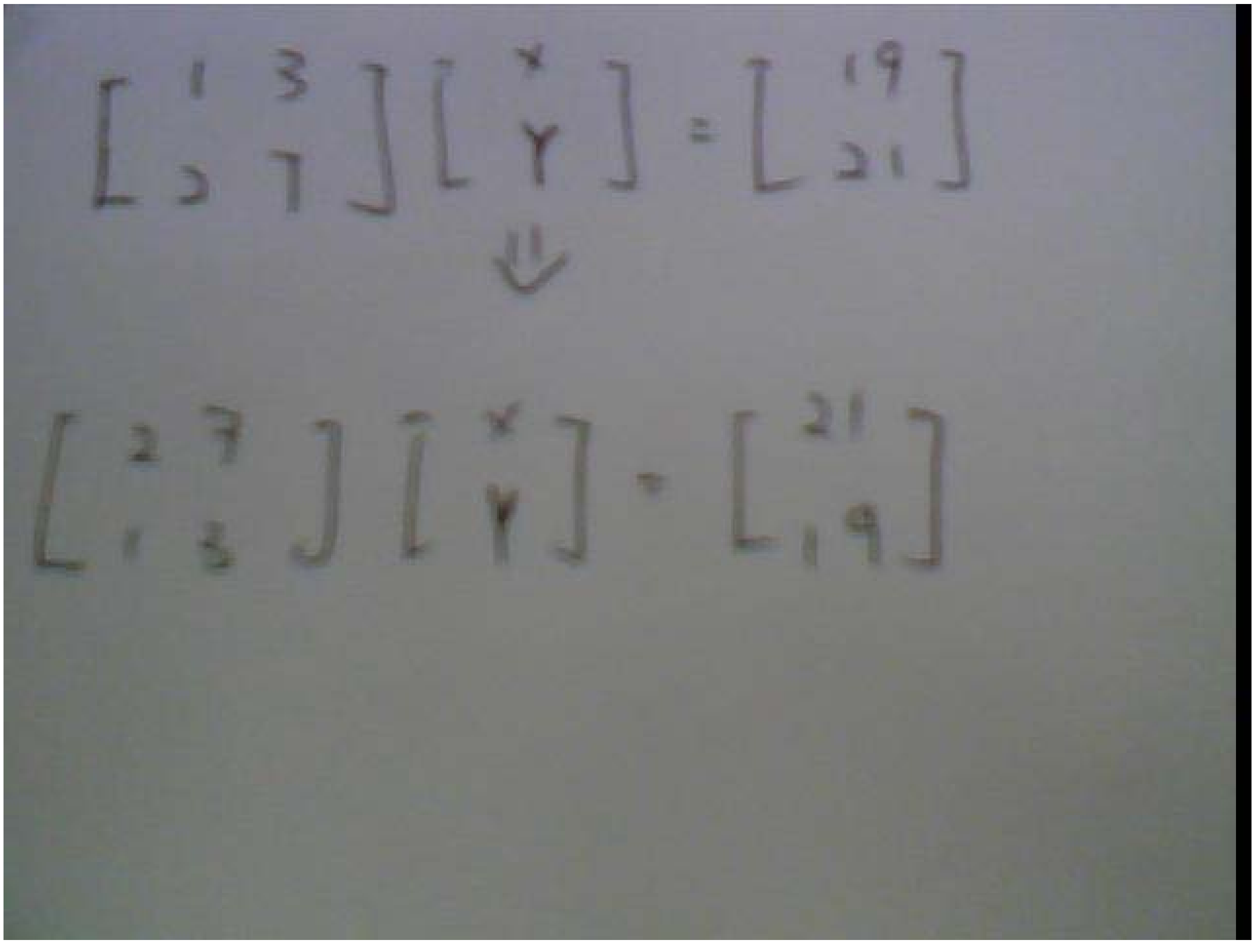
With Guass elimination we can convert an arbitrary system of linear equation into upper triangular and solve with backward substitution.

Guass elimination uses some elementary transformations of the system of equations that do not affect the solution

1. Interchange
2. Scaling
3. replacement

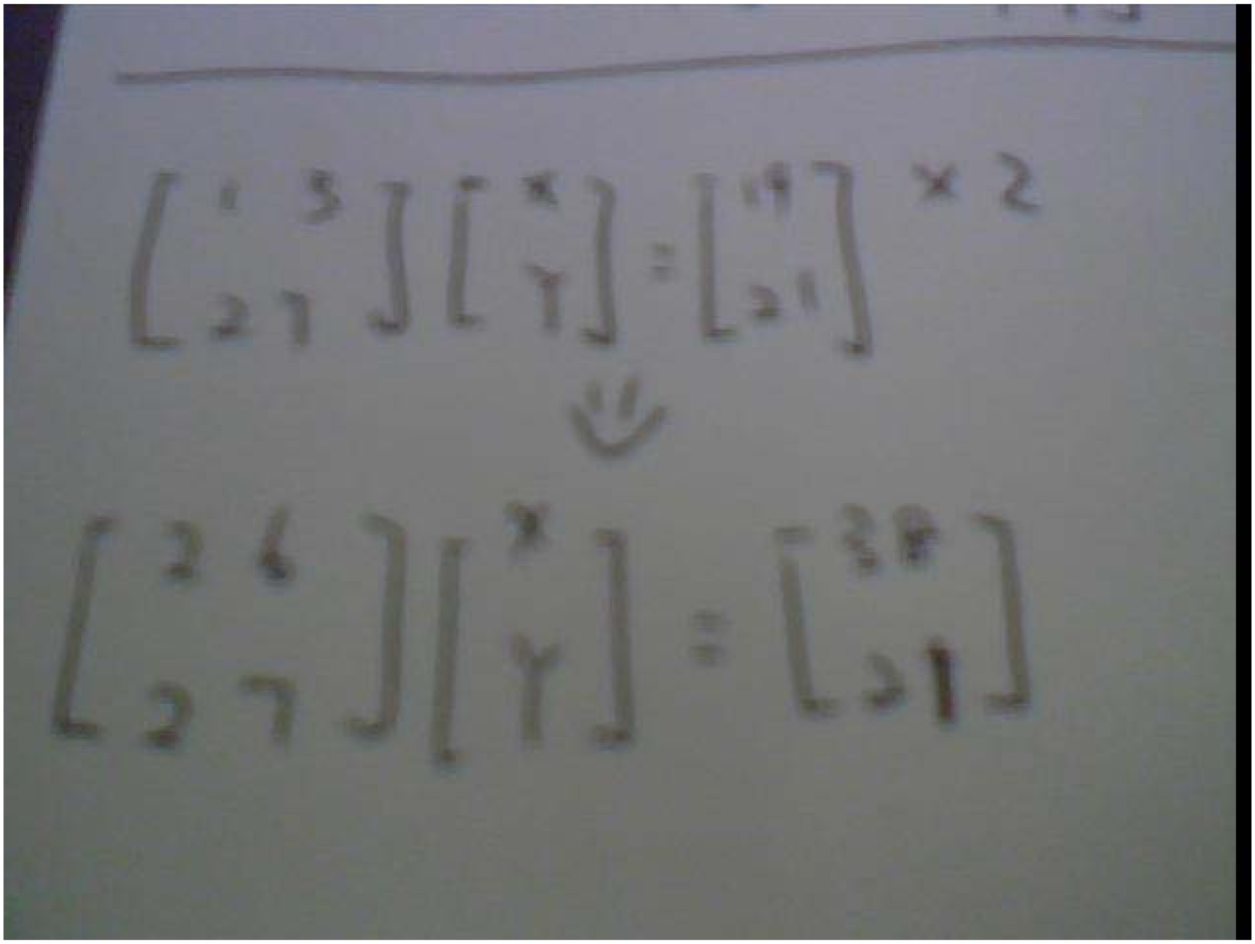
Interchange

The order of two equations can be change and it will not affect the solution



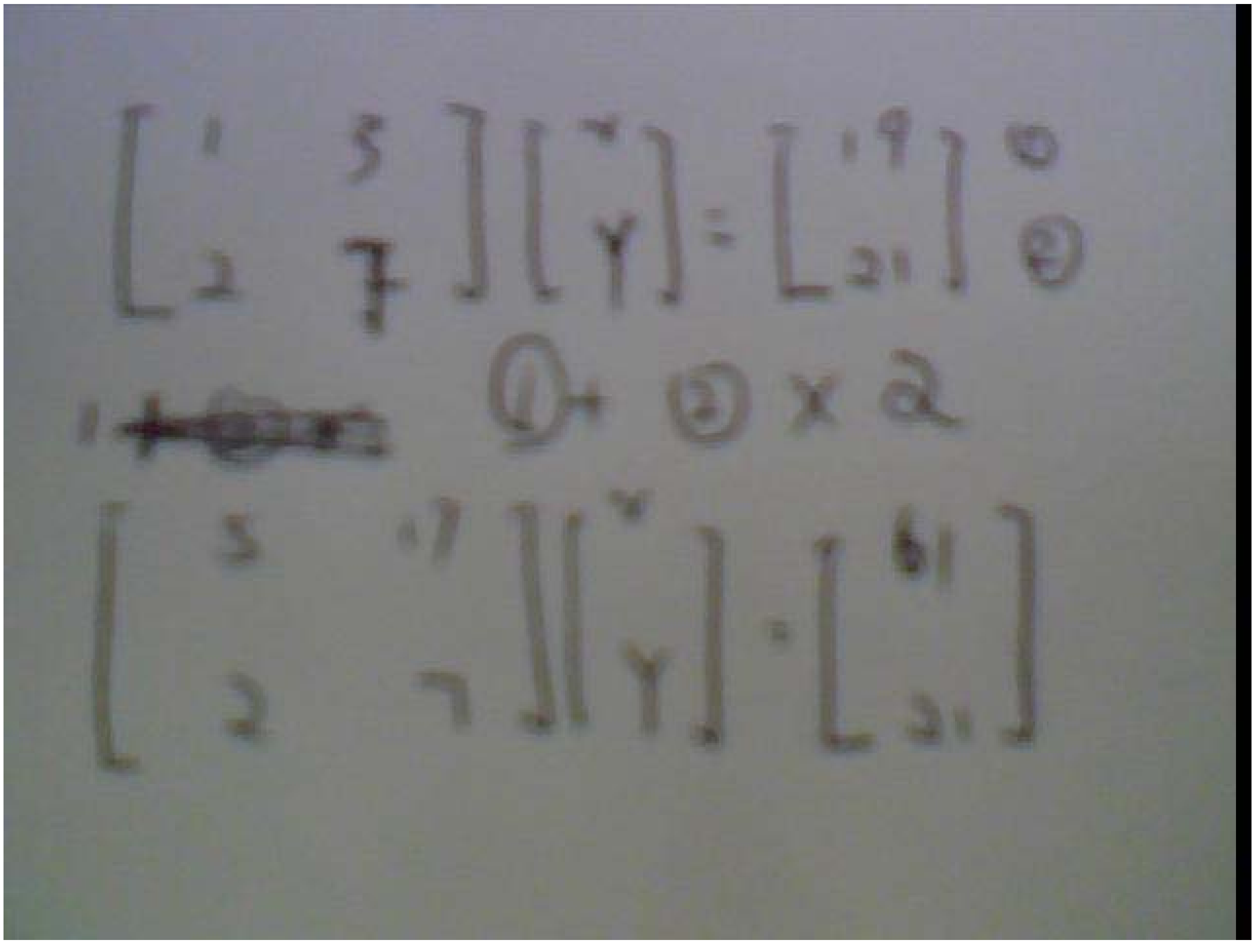
<u>Scaling</u>

Multiplying an equation by a constant will not affect the solution

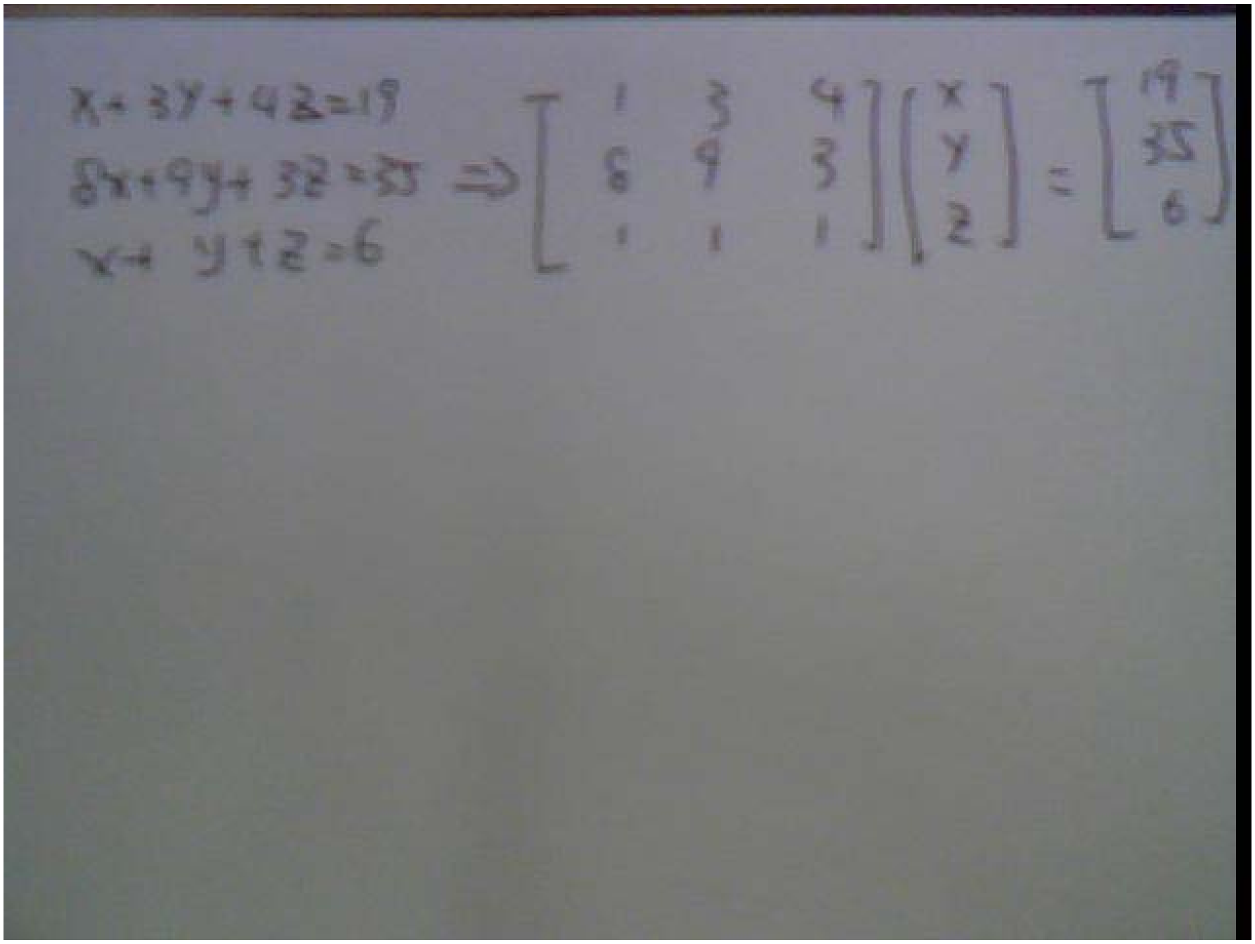


<u>Replacement</u>

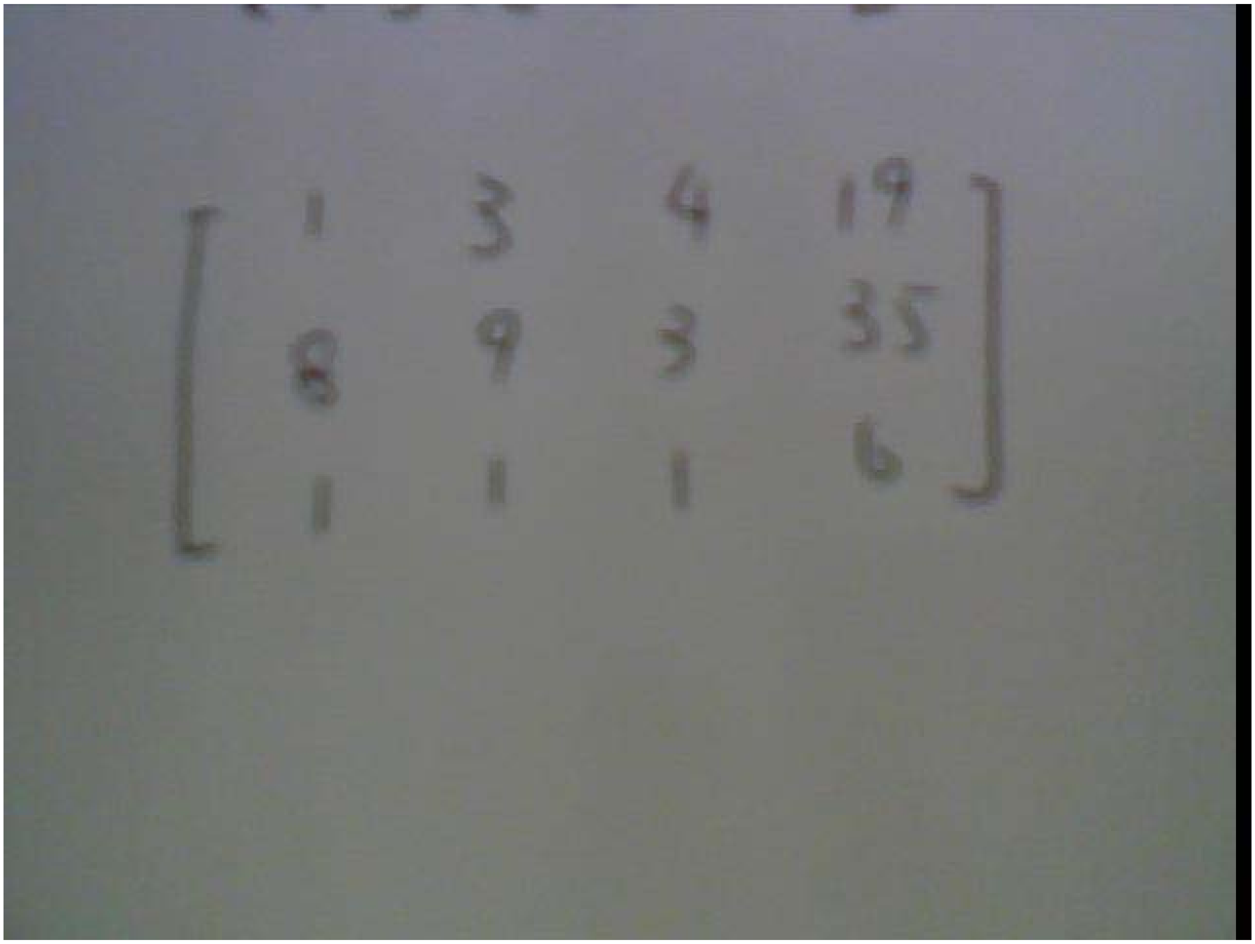
A equation can be replaced by the sum of itself and a non-zero multiple of another equation without affecting the solution



We used these transformations to convert a system of linear equations to upper triangular and then solve it with backward substitution



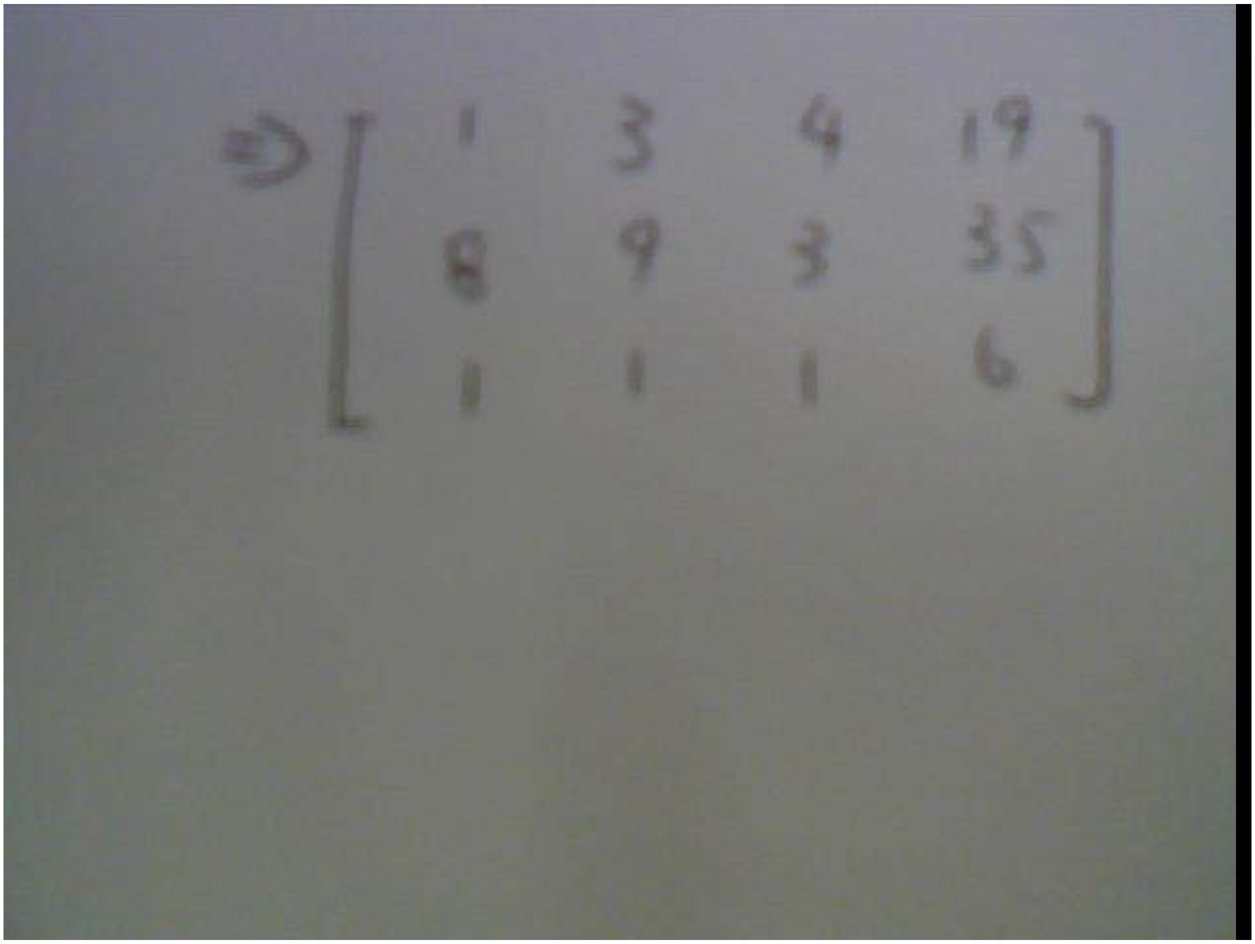
Augment matrix



For all the row's starting at row=i=1

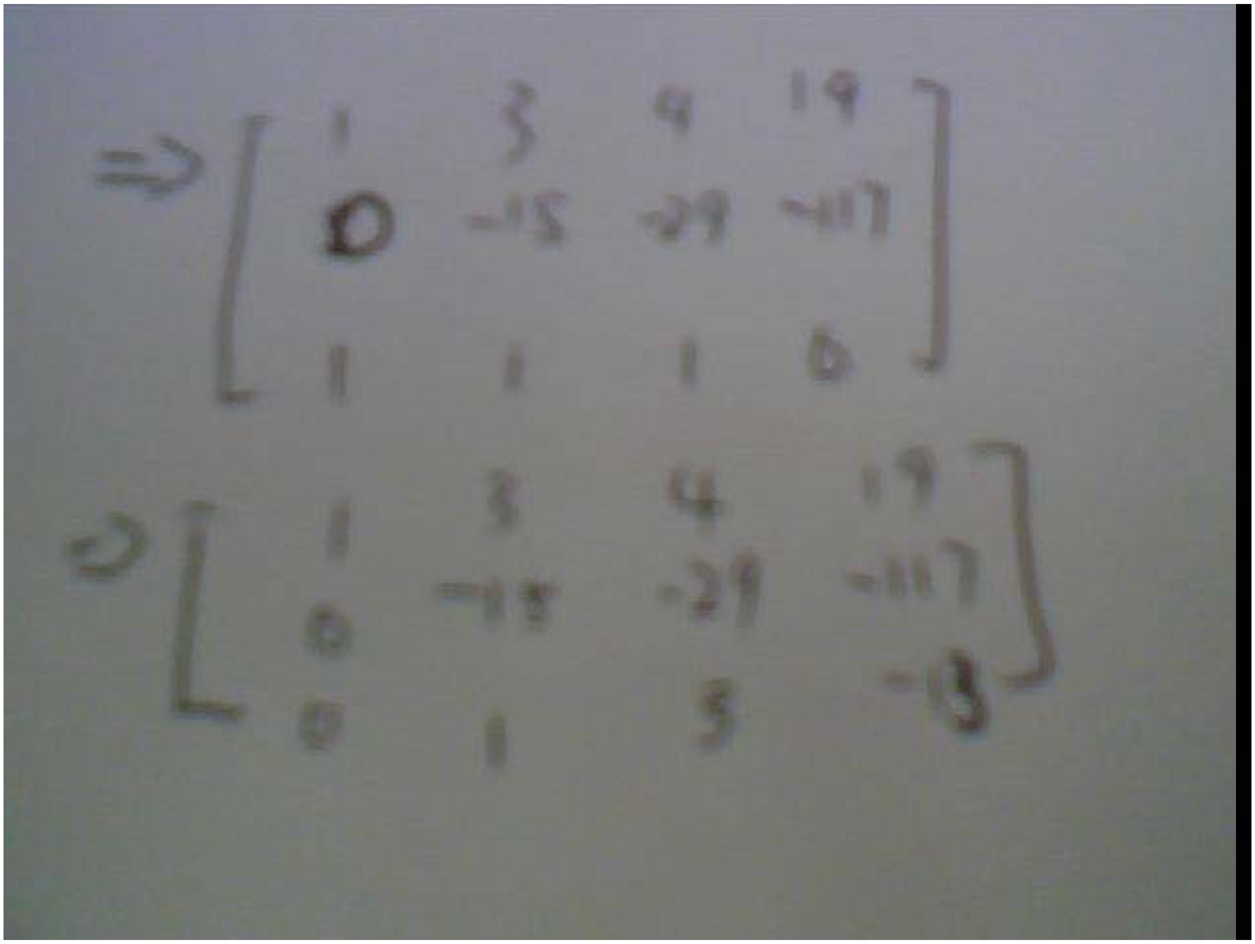
Step one:

Name element aii as the pivot, aii, we have to make sure that the pivot is 1, otherwise divide the whole row by the value of the pivot.



Step two

We need to make the matrix upper triangular we need to make the element under the pivot 0, we multiply the pivot by the –value under the pivot and added the row.

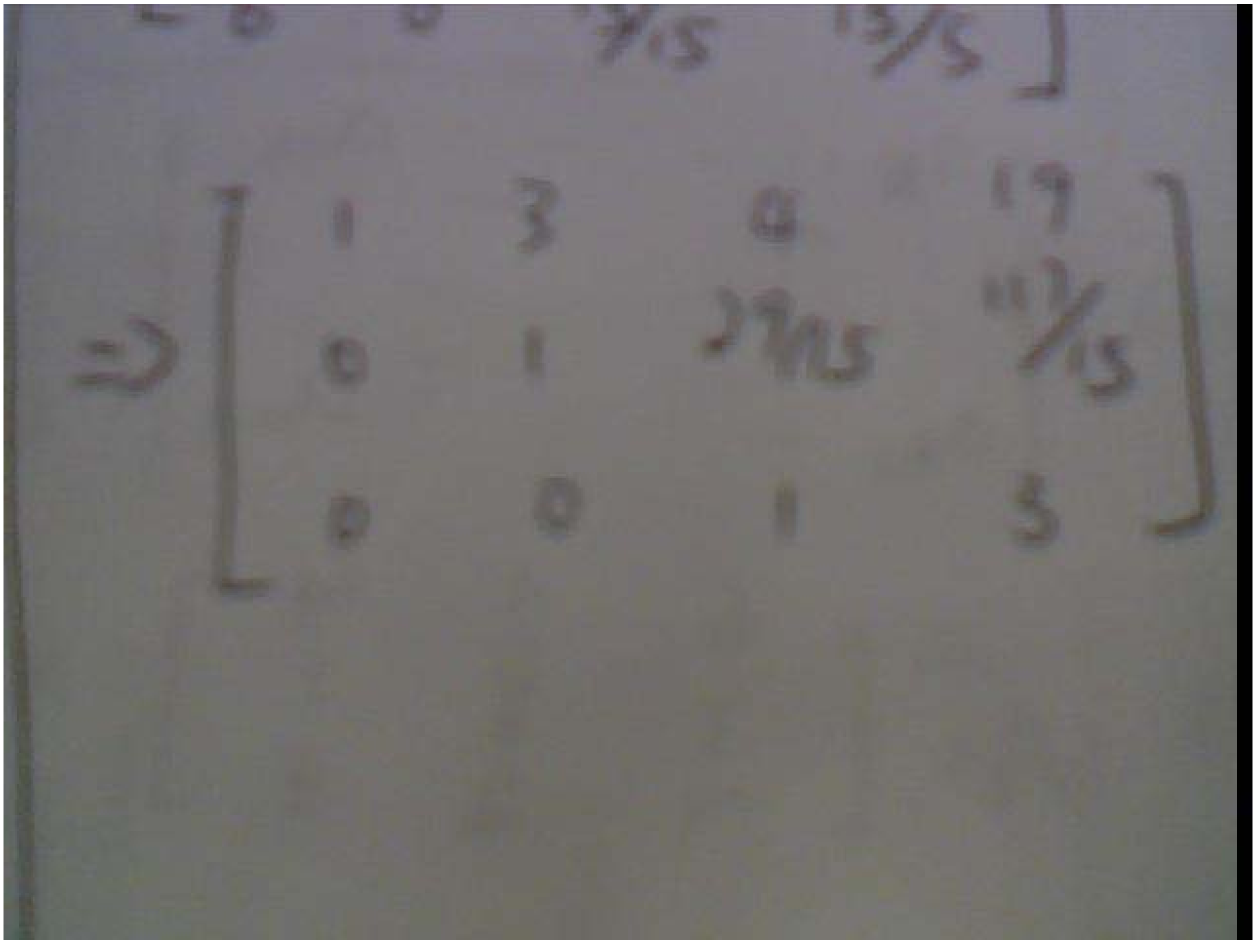


Step three

Now the pivot is $a_{22}=-15$, repeat step one and two

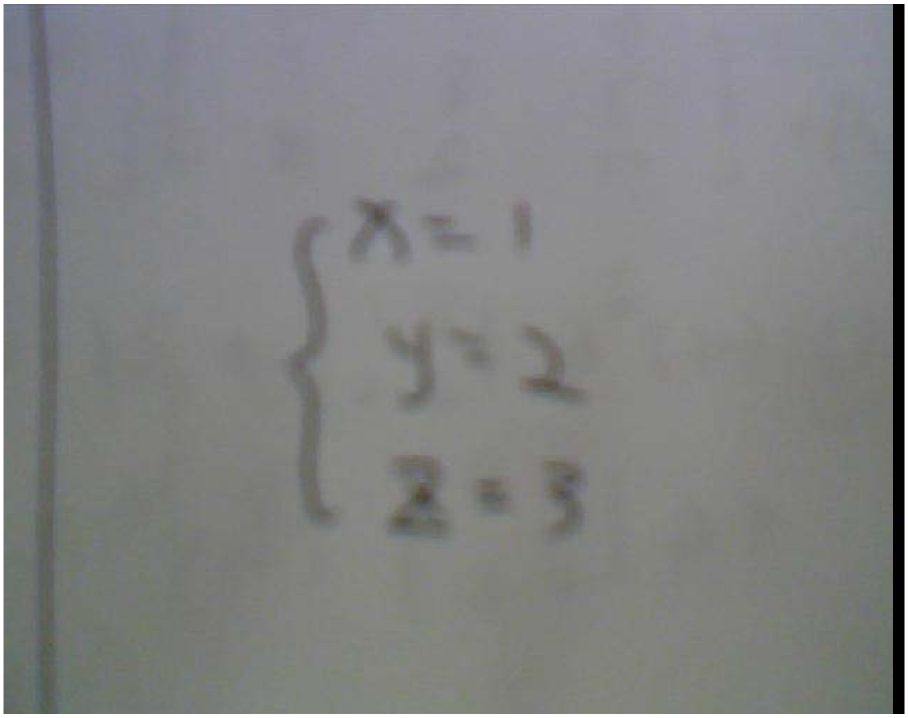
$$\Rightarrow \begin{bmatrix} 1 & 3 & 4 & 19 \\ 0 & 1 & \frac{29}{15} & \frac{117}{15} \\ 0 & -2 & -3 & -15 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 3 & 4 & 19 \\ 0 & 1 & \frac{29}{15} & 117/15 \\ 0 & 0 & 13/15 & 13/5 \end{bmatrix}$$



Step four

Backward substitution



Note: if the pivot is 0, then swap rows

Implementing Guass Elimination to Matlab

```
File gauss.m
function x = gauss(A,B)
% Input –   A is a NxN matrix
%           B is a Nx1 matrix
% Output – x is a Nx1 matrix with the solution Ax=B
% Get the dimensions of A
[N N] = size(A)
% Initialize the x vector with 0
x = zero(N,1)
% Construct the augmented Matrix
Aug = [A B]
%Gaussian Elimination
% Pivot all rows
for p = 1:N
% get pivot to simplify doesn't include code where pivot is 0 you will have to swap rows
piv = Aug[p,p]
```