The system of Linear Equations

Introduction

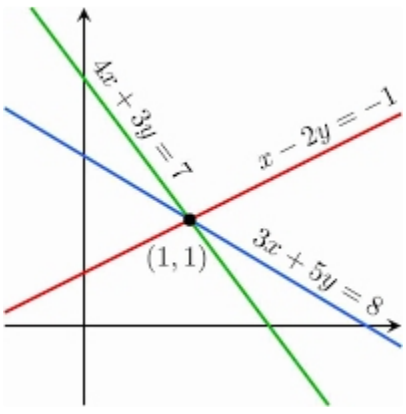
We have equations having the maximum degree of 1 and generally, the total number of equations is same as unknown variables.
In general system looks like,

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1n}x_n = b_1$
$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ... + a_{2n}x_n = b_2$
...
$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + ... + a_{nn}x_n = b_n$

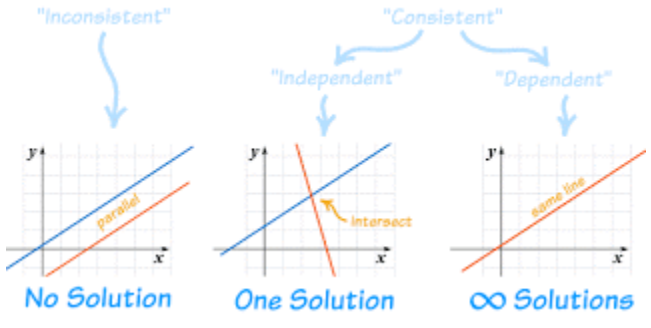
Matrix representation

$AX = B$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & .. & a_{2n} \\ a_{31} & a_{32} & a_{33} & ... & a_{3n} \\ ... & .... & ... & .... & | \\ a_{n1} & a_{n2} & a_{n3} & ... & a_{nn} \end{vmatrix} * \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_n \end{vmatrix} = \begin{vmatrix} b_1 \\ b_2 \\ b_3 \\ .... \\ b_5 \end{vmatrix}$$



It is recommended to read about the consistency of linear equations system, whether the system has a solution or not or have infinite solutions.



Also go through the matrix properties and operation like the rank of a matrix, echelon form and different types of matrix.

Jacobi's Method

posted Dec 28, 2017, 10:59 AM by Atul Rana **[updated 36 minutes ago]**

Description

An iterative method of the solving system of linear equation. The amount of computation depends upon the degree of accuracy required, for a large system of equation iterative method is faster than direct methods.

we have a system

$a_1x + b_1y + c_1z = d_1$
$a_2x + b_2y + c_2z = d_2$
$a_3x + b_3y + c_3z = d_3$

first we write equation as below

$x = 1/a_1(d_1 - b_1y - c_1z)$
$y = 1/b_2(d_2 - a_2x - c_2z)$
$z = 1/c_3(d_3 - a_3x - b_3y)$ **a_1, b_2, c_3 usually the largest cofficent in the corresponding equations.**

x_0, y_0 ,z_0 are initial approximation in the absence of initial approximation use x_0 =0, y_0=0, z_0=0;

by substituting

$x_1 = 1/a_1(d_1 - b_1y_0 - c_1z_0)$

$y_1 = 1/b_2(d_2 - a_2x_0 - c_2z_0)$

$z_1 = 1/c_3(d_3 - a_3x_0 - b_3y_0)$

repeat the same process for more accurate approximations

x_2, y_2, z_2 x_3, y_3, z_3 and so on.

- **Programming implementation of Jacobi's Method in C++**

```
#include<iostream>
#include<math.h>
using namespace std;

int main(){

    double a1, b1, c1,  x, d1;
    double a2, b2, c2,  y, d2;
    double a3, b3, c3,  z, d3;
    //                AX = D

    cout<<"EQuation 1: a1x + b1y + c1z = d1 (a1 != 0)"<<endl;
    cin>>a1>>b1>>c1>>d1;
    cout<<"EQuation 2: a2x + b2y + c2z = d2 (b2 != 0)"<<endl;
    cin>>a2>>b2>>c2>>d2;
    cout<<"EQuation 3: a3x + b3y + c3z = d3 (c3 != 0)"<<endl;
    cin>>a3>>b3>>c3>>d3;


    char ch='n';
    double xpre=0,ypre=0,zpre=0;

    cout<<"Want to provide initial values of x,y,z (y/n):";
    cin>>ch;
    if(ch == 'y'){
        cout<<"Give initial values of x,y,z"<<endl;
        cin>>xpre>>ypre>>zpre;
    }

    for(int i=0;i<10;i++){
        x = (d1 - b1*ypre - c1*zpre)*(1/a1);
        y = (d2 - a2*xpre - c2*zpre)*(1/b2);
        z = (d3 - a3*xpre - b3*ypre)*(1/c3);

        x = round(x*10000.0)/10000.0;
        y = round(y*10000.0)/10000.0;
        z = round(z*10000.0)/10000.0;
        xpre = x; ypre = y; zpre = z;
        cout<<x<<" "<<y<<" "<<z<<endl;
    }
    printf("\nValues of (x,y,z) correct upto 4 decimal places: %0.4f %0.4f
%0.4f\n",x,y,z);
    return 0;
}
```

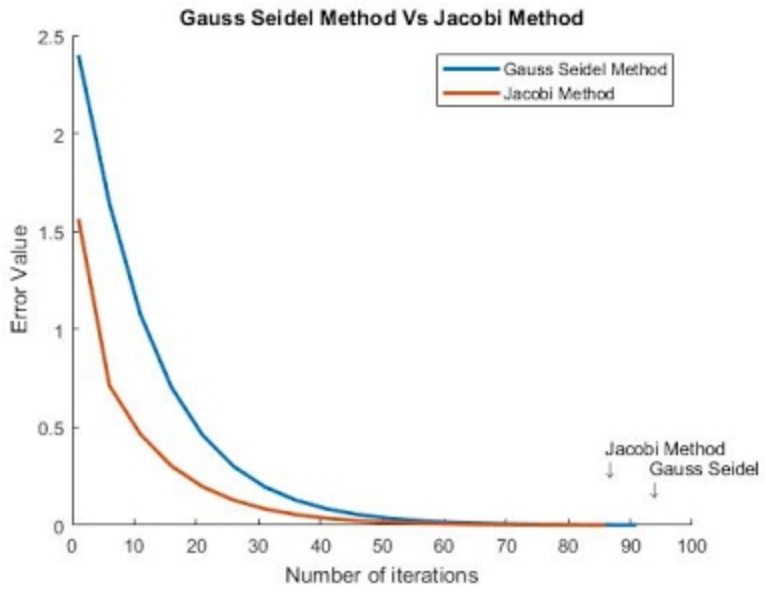
- **Code execution**

```
EQuation 1: a1x + b1y + c1z = d1 (a1 != 0)
27 6 -1 85
EQuation 2: a2x + b2y + c2z = d2 (b2 != 0)
6 15 2 72
EQuation 3: a3x + b3y + c3z = d3 (c3 != 0)
1 1 54 110
Want to provide initial values of x,y,z (y/n):n
3.1481 4.8 2.037
2.1569 3.2692 1.8899
2.4917 3.6853 1.9366
2.4009 3.5451 1.9226
2.4316 3.5833 1.9269
2.4232 3.5704 1.9257
2.426 3.574 1.926
2.4253 3.5728 1.9259
2.4255 3.5731 1.926
2.4255 3.573 1.926

Values of (x,y,z) correct upto 4 decimal places: 2.4255 3.5730 1.9260
```

Note: a_1,b_2,c_3 has to be the largest coefficient in the corresponding equation unless method never converges to approximate value of the unknown variable, try this code for the different system of equation and observe the output of the code.

- **Observation**



Gauss Seidel Method

posted Dec 28, 2017, 10:58 AM by Atul Rana **[updated 37 minutes ago]**

Description

Gauss-Seidel Method is just the little bit different than Jacobi's method but converges to the solution way less iteration than the Jacobi's.We have a system of linear equations

$a_1x + b_1y + c_1z = d_1$

$a_2x + b_2y + c_2z = d_2$

$a_3x + b_3y + c_3z = d_3$

first we write equation as below

$x = 1/a_1(d_1 - b_1y - c_1z)$

$y = 1/b_2(d_2 - a_2x - c_2z)$

$z = 1/c_3(d_3 - a_3x - b_3y)$

If x_0, y_0 ,z_0 are initial approximation, in the absence of initial approximation use x_0 =0, y_0=0, z_0=0;
by substituting

$x_1 = 1/a_1(d_1 - b_1y_0 - c_1z_0)$

$y_1 = 1/b_2(d_2 - a_2x_1 - c_2z_0)$

$z_1 = 1/c_3(d_3 - a_3x_1 - b_3y_1)$ notice we are using the updated values of variables repeat the same process to get more accurate result.

- **Programming implementation of Gauss-Seidel Method in C++**

```
#include<iostream>
using namespace std;

int main(){

    double a1, b1, c1,  x, d1;
    double a2, b2, c2,  y, d2;
    double a3, b3, c3,  z, d3;
    //                AX = D

    cout<<"EQuation 1: a1x + b1y + c1z = d1 (a1 != 0)"<<endl;
    cin>>a1>>b1>>c1>>d1;
    cout<<"EQuation 2: a2x + b2y + c2z = d2 (b2 != 0)"<<endl;
    cin>>a2>>b2>>c2>>d2;
    cout<<"EQuation 3: a3x + b3y + c3z = d3 (c3 != 0)"<<endl;
    cin>>a3>>b3>>c3>>d3;

    char ch;
    cout<<"Want to provide initial values of x,y,z (y/n):";
    cin>>ch;

    if(ch == 'y'){
        cout<<"Give initial values of x,y,z"<<endl;
        cin>>x>>y>>z;
    }
    else{
        x=0;y=0;z=0;
    }
    for(int i=0;i<15;i++){
        x = (d1 - b1*y - c1*z)/a1;
        y = (d2 - a2*x - c2*z)/b2;
        z = (d3 - a3*x - b3*y)/c3;
        cout<<x<<" "<<y<<" "<<z<<endl;
    }

    printf("\nValues of (x,y,z) correct upto 4 decimal places: %0.4f %0.4f
%0.4f\n",x,y,z);
    return 0;
```

```
}
```

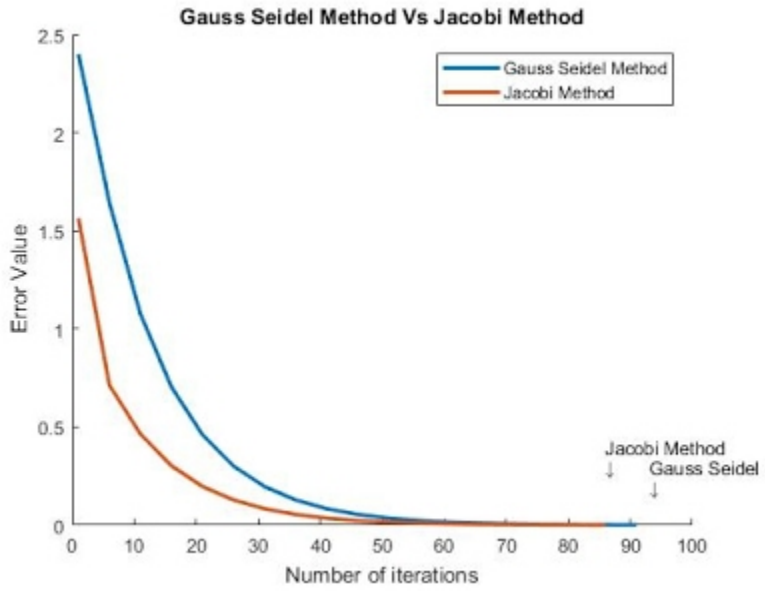
- **Code execution**

```
EQuation 1: a1x + b1y + c1z = d1 (a1 != 0)
5 -4 1 10
EQuation 2: a2x + b2y + c2z = d2 (b2 != 0)
2 4 0 12
EQuation 3: a3x + b3y + c3z = d3 (c3 != 0)
1 4 5 -1
Want to provide initial values of x,y,z (y/n):n
2 2 -2.2
4.04 0.98 -1.792
3.1424 1.4288 -1.97152
3.53734 1.23133 -1.89253
3.36357 1.31822 -1.92729
3.44003 1.27999 -1.91199
3.40639 1.29681 -1.91872
3.42119 1.28941 -1.91576
3.41468 1.29266 -1.91706
3.41754 1.29123 -1.91649
3.41628 1.29186 -1.91674
3.41684 1.29158 -1.91663
3.41659 1.2917 -1.91668
3.4167 1.29165 -1.91666
3.41665 1.29167 -1.91667

Values of (x,y,z) correct upto 4 decimal places: 3.4167 1.2917 -1.9167
```

Note: a_1,b_2,c_3 must be the largest coefficient in the corresponding equation unless solution may not converges to approximate values of the unknown variables.

- **Observation**



LU Decomposition Method(Cholesky's Decomposition)

posted Dec 28, 2017, 10:57 AM by Atul Rana **[updated 18 minutes ago]**

Description

Consider the following these linear equation in three variables

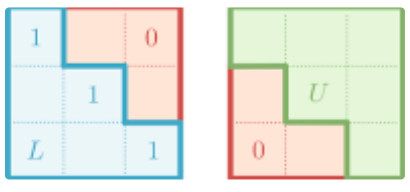
$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

So in matrix form AX = B

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} * \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix} = \begin{vmatrix} b_1 \\ b_2 \\ b_3 \end{vmatrix}$$

Let A = LU

LU = A



$$\begin{vmatrix} 1 & 0 & 0 \\ & & \\ & & \end{vmatrix} \begin{vmatrix} u_{11} & u_{12} & u_{13} \\ & & \\ & & \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ & & \\ & & \end{vmatrix}$$

$$\begin{vmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{vmatrix} * \begin{vmatrix} 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{vmatrix} = \begin{vmatrix} a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

substituting A

LUX = B

LV = B where UX = V

first, we find the element values of matrix L and U which is just simple matrix multiplication and compare the resultant corresponding equation with the values in A.

and then using matrix B and L find V, after use V and U to find out the matrix X.That's it.

- **Programming implementation of LU Decomposition Method in C++.**

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    float a11,a12,a13, b1;
    float a21,a22,a23, b2;
    float a31,a32,a33, b3;

    float            u11, u12, u13;
    float l21,            u22, u23;
    float l31, l32,            u33;

    float x1,    v1;
    float x2,    v2;
    float x3,    v3;


    cout<<"a11x1 + a12x2 + a13x3 = b1"<<endl;
    cout<<"give values of: a11 a12 a13 b1"<<endl;
    cin>>a11>>a12>>a13>>b1;

    cout<<"a21x1 + a22x2 + a23x3 = b2"<<endl;
    cout<<"give values of: a21 a22 a23 b2"<<endl;
    cin>>a21>>a22>>a23>>b2;

    cout<<"a31x1 + a32x2 + a33x3 = b3"<<endl;
    cout<<"give values of: a31 a32 a33 b3"<<endl;
    cin>>a31>>a32>>a33>>b3;


    u11 = a11;
    u12 = a12;
    u13 = a13;
    l21 = a21 / u11;
    u22 = a22 - l21*u12;
    u23 = a23 - l21*u13;
    l31 = a31 / u11;
    l32 = (a32 - l31*u12) / u22;
    u33 = a33 - l31*u13 - l32*u23;
/*
    LU = A
    |    1    0    0 ||  u11    u12       u13 |   | a11  a12  a13 |
    |    l21  1    0 |*|    0    u22       u23 | = | a21  a22  a23 |
    |    l31 l32   1 ||    0      0       u33 |   | a31  a32  a33 |
*/
    //  LUX = B
    //  LV = B
    // we the values of matrix L
    v1 = b1;
    v2 = b2 - l21*v1;
    v3 = b3 - l31*v1 - l32*v2;

    //    UX = V
    //Now we now the values of Matrix V as well U;

    x3 = v3 / u33;
    x2 = (v2 - u23*x3) / u22;
    x1 = (v1 - u13*x3 - u12*x2) / u11;

    cout<<"Values of x1 ,x2, x3:"<<endl;
    cout<<x1<<endl;
    cout<<x2<<endl;
    cout<<x3<<endl;
    return 0;
}
```

- **Code execution**

```
a11x1 + a12x2 + a13x3 = b1
give values of: a11 a12 a13 b1
1 1 1 9
a21x1 + a22x2 + a23x3 = b2
give values of: a21 a22 a23 b2
2 -3 4 13
a31x1 + a32x2 + a33x3 = b3
give values of: a31 a32 a33 b3
3 4 5 40
Values of x1 ,x2, x3:
1
3
5
```