Jacobi Iteration

Theory

The Jacobian Method, also known as the Jacobi Iterative Method, is a fundamental algorithm used to solve systems of linear equations. It is useful when dealing with large systems where direct methods (like Gaussian elimination) are computationally expensive.

The Jacobi iterative method is a specific implementation of the Jacobian method. It assumes that the system of linear equations can be written in the form Ax = b, where A is the coefficient matrix, x is the vector of unknown variables, and b is the vector of constants. The Jacobi method proceeds as follows:

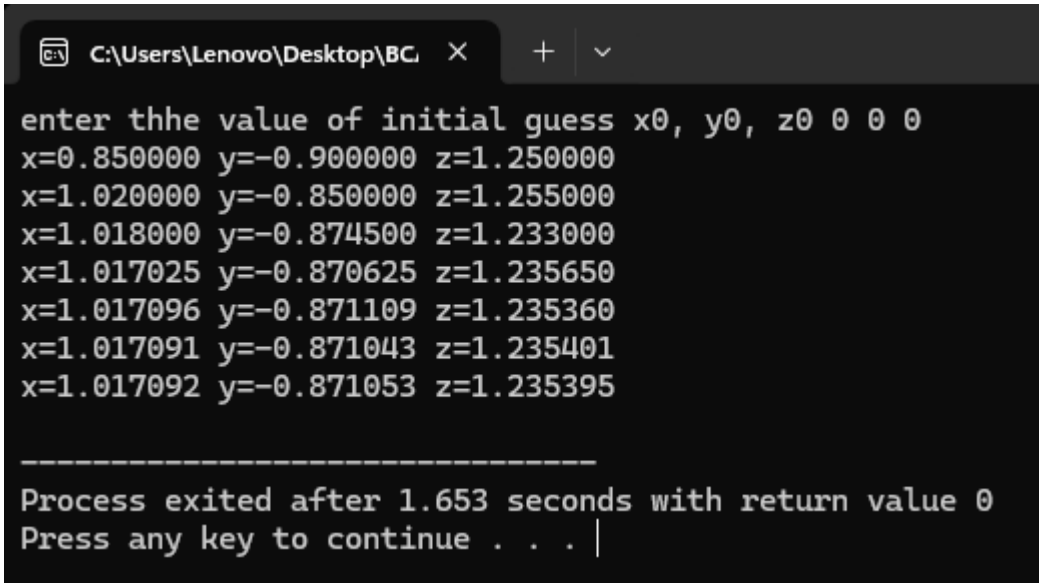
- Assumption 1: The system of linear equations has a unique solution.
- Assumption 2: The coefficient matrix A has no zeros on its main diagonal.
- Assumption 3: The system is diagonally doominant(i.e., the absolute value of the diagonal elemnet in each row must be greater than or equal to the sum of the absolute values of the other elements in that row.)

Code

```
//jacobi-iteration

/* solve
2x+y-2z = 17
3x + 20y - z = -18
2x + 3y + 20z = 25 */

#include<stdio.h>
#include<math.h>


#define f1(x,y,z) (17-y+2*z)/20
#define f2(x,y,z) (-18-2*x-3*y)/20
#define f3(x,y,z) (25-2*x-2*y)/20

#define e 0.000001

int main(){

	float x0, y0, z0, x1, y1, z1, e1, e2, e3;

	printf("enter thhe value of initial guess x0, y0, z0");
	scanf("%f %f %f", &x0, &y0, &z0);

	do{
		x1 = f1(x0,y0,z0);
		y1 = f2(x0,y0,z0);
		z1 = f3(x0,y0,z0);
		e1 = fabs(x1-x0);
		e2 = fabs(y1-y0);
```

```
                e3 = fabs(z1-z0);
                x0 = x1;
                y0 = y1;
                z0 = z1;
                printf("x=%f y=%f z=%f", x1, y1, z1);
        }while (e1>e & e2>e & e3>e);
}
```

Output



Gauss-Seidel Iteration

Theory

The Gauss-Seidel method is an iterative technique used to solve a square system of linear equations. It is a popular method in numerical linear algebra due to its simplicity and efficiency.

Ensure matrix is diagonally dominant, choose initial solution $x^{(0)}$,For each equation in the system use updated values of x_1, x_2, ..., x_{i-} and use previous values of x_{i+1}, ..., x_n to calculate new x_i. Continue until convergence or until maximum iterations reached.

Code

```
//Gauss Seidel iteration

/* solve
2x+y-2z = 17
3x + 20y - z = -18
2x + 3y + 20z = 25 */

#include <stdio.h>
#include <math.h>

#define f1(y, z) (17 - y + 2 * z) / 20
#define f2(x, z) (-18 - 2 * x - 3 * z) / 20
#define f3(x, y) (25 - 2 * x - 2 * y) / 20

#define e 0.000001

int main() {
   float x0, y0, z0, x1, y1, z1, e1, e2, e3;

   printf("Enter the initial guesses for x0, y0, z0: ");
   scanf("%f %f %f", &x0, &y0, &z0);

   do {
      x1 = f1(y0, z0);
      y1 = f2(x1, z0);
      z1 = f3(x1, y1);

      e1 = fabs(x1 - x0);
      e2 = fabs(y1 - y0);
      e3 = fabs(z1 - z0);

      x0 = x1;
      y0 = y1;
      z0 = z1;

      printf("x = %f, y = %f, z = %f\n", x1, y1, z1);
   } while (e1 > e || e2 > e || e3 > e);
```

```
    return 0;
}
```

Output