Indian Institute of Information Technology, Vadodara

Parallel Programming(CS403)

Prof. Reshmi Mitra

Name - Dilip Puri
ID - 201351014
Collaborators - Hemant Kumar(201352026)
Govind Meena(201352010)

Project 1

Submission Date - September 12, 2016 **Deadline** - Sep12, 11.59 PM

1 Introduction

Let A be a square matrix. If there is a lower triangular matrix L with all diagonal entries equal to 1 and an upper matrix U such that A=LU, then we say that A has an LU-decomposition. It can be helpful in calculating various types of operation on matrices. Here L matrix has the upper triangular values as 0 and U has lower triangular values as 0 and diagonal values same as that of the original matrix.

2 Algorithm

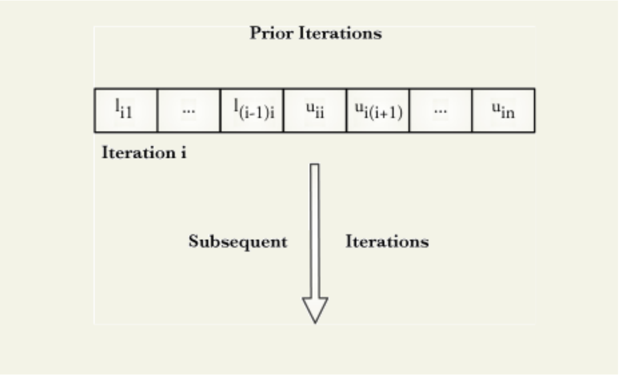
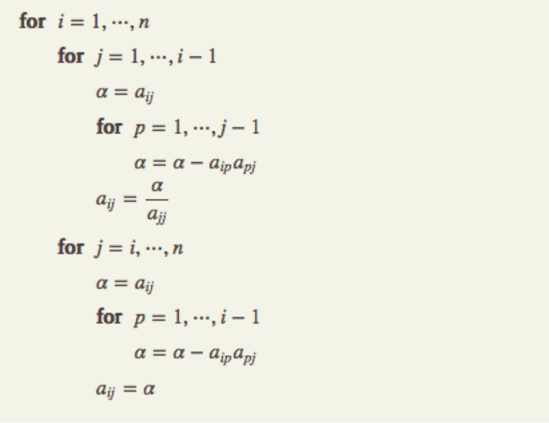


Figure 1: Computational Sequence of Doolittle's Method



```
for $i = 1, \cdots, n$
    for $j = 1, \cdots, i-1$
        $\alpha = a_{ij}$
        for $p = 1, \cdots, j-1$
            $\alpha = \alpha - a_{ip}a_{pj}$
        $a_{ij} = \frac{\alpha}{a_{jj}}$
    for $j = i, \cdots, n$
        $\alpha = a_{ij}$
        for $p = 1, \cdots, i-1$
            $\alpha = \alpha - a_{ip}a_{pj}$
        $a_{ij} = \alpha$
```

Figure 2: Doolittle's LU Decompostion Algorithm

3 Serial Code

Listing 1: Code

```
#include<stdio.h>

int main(void){

    int i,j,k,n;
    printf("Enter the order of square matrix: ");
    scanf("%d",&n);

    float A[n][n],L[n][n], U[n][n];

    printf("Enter matrix element:\n");

    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
        {
            printf("Enter A[%d][%d] element: ", i,j);
            scanf("%f",&A[i][j]);
        }
    }

    for(j=0; j<n; j++)
    {
        for(i=0; i<n; i++)
        {
            if(i<=j)
            {
                U[i][j]=A[i][j];
                for(k=0; k<=i-1; k++)
                    U[i][j]-=L[i][k]*U[k][j];
                if(i==j)
                    L[i][j]=1;
                else
                    L[i][j]=0;
            }
            else
            {
                L[i][j]=A[i][j];
                for(k=0; k<=j-1; k++)
                    L[i][j]-=L[i][k]*U[k][j];
                L[i][j]/=U[j][j];
                U[i][j]=0;
            }
        }
    }

    printf("[L]: \n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf("%9.3f",L[i][j]);
        printf("\n");
    }
    printf("\n\n[U]: \n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf("%9.3f",U[i][j]);
        printf("\n");
    }

return 0;
}
```

4 Analysis using Valgrind